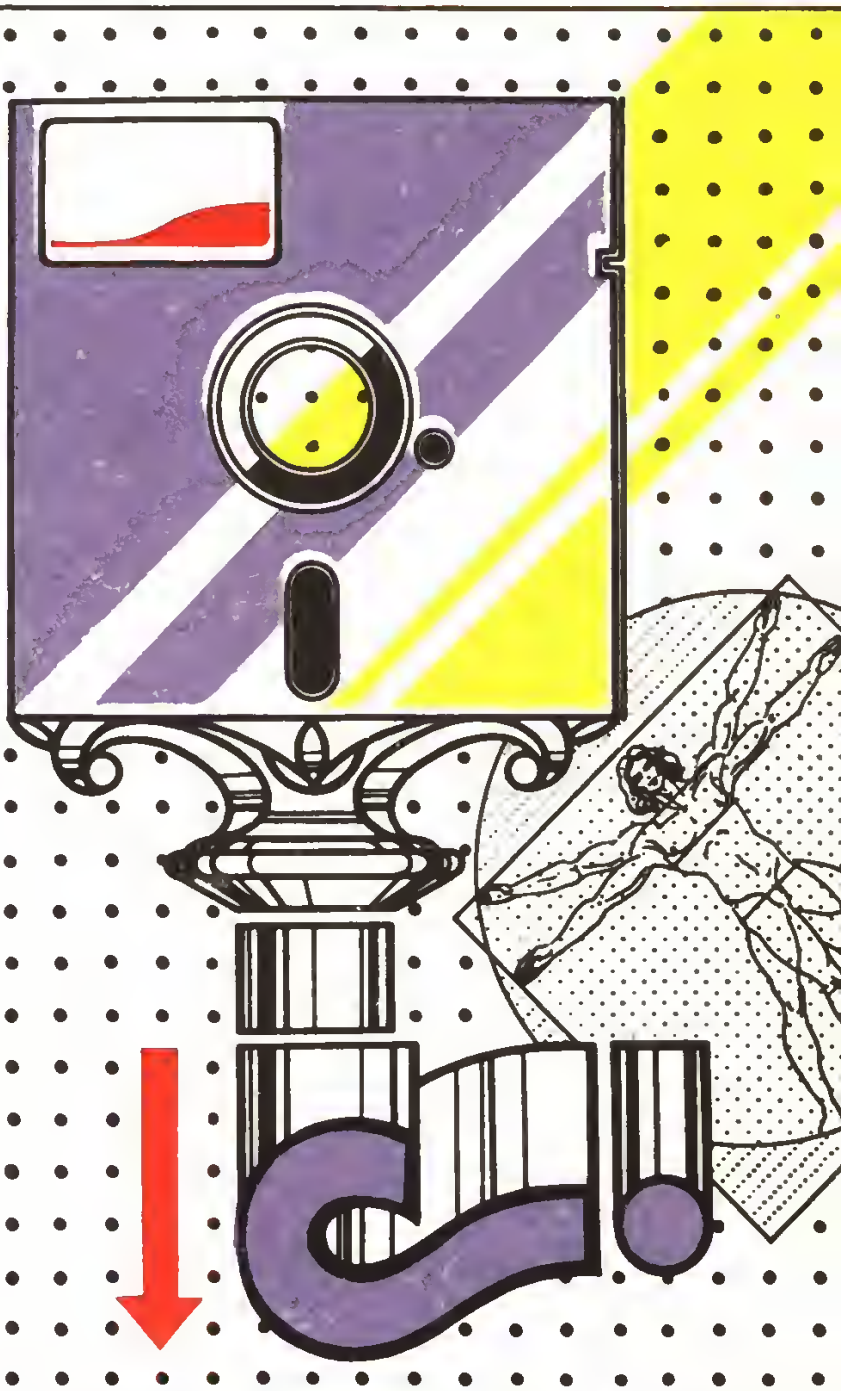
Б. Сойер, Д. Л. Фостер

ПРОГРАММИРОВАНИЕ ЭКСПЕРТНЫХ СИСТЕМ НА ПАСКАЛЕ



ПРОГРАММИРОВАНИЕ ЭКСПЕРТНЫХ СИСТЕМ НА ПАСКАЛЕ

PROGRAMMING EXPERT SYSTEMS IN PASCAL

BRIAN SAWYER
DENNIS FOSTER

A WILEY PRESS BOOK
JOHN WILEY & SONS, INC.
NEW YORK · CHICHESTER · BRISBANE · TORONTO · SINGAPORE

Б. Сойер
Д. Л. Фостер

ПРОГРАММИРОВАНИЕ ЭКСПЕРТНЫХ СИСТЕМ НА ПАСКАЛЕ

Перевод с английского
В. А. Белова
Предисловие чл.-кор. АН СССР
В. П. Иванникова

МОСКВА
"ФИНАНСЫ И СТАТИСТИКА"
1990

ББК 24.4.1
С54

Сойер Б., Фостер Д.Л.

С54 Программирование экспертных систем на Паскале: Пер с англ.; Предисловие В.П.Иванникова.- М.: Финансы и статистика, 1990. -191 с.:ил.
ISBN 5-279-00449-9.

Книга американских авторов рассчитана на программистов, желающих создавать собственные экспертные системы. Описаны архитектура и процесс создания небольшой продукционной системы на языке Турбо Паскаль с обратной последовательностью вывода. Система документирована полными листингами программ и примерами небольших баз знаний.

Рассмотрен интерфейс ЭС с файлами СУБД и электронных таблиц.

Для программистов, практически осваивающих технологии искусственного интеллекта.

2404010000 – 123
С————————129 -- 90
010(01) -- 90

ББК 24.4.1

ISBN 5-279-00449-9 (СССР)

ISBN 0-471-84267-2 (США)

ПРЕДИСЛОВИЕ К РУССКОМУ ИЗДАНИЮ

Экспертные системы. Наверное, в настоящее время нет человека, занимающегося или просто интересующегося информатикой, которому не было бы знакомо это словосочетание. Экспертные системы (ЭС) составляют часть систем искусственного интеллекта (ИИ), причем самую существенную их часть. Экспертные системы обычно определяют как программы ЭВМ, моделирующие действия эксперта-человека при решении задач в узкой предметной области, на основе накопленных знаний, составляющих базу знаний (БЗ).

Термин "искусственный интеллект" был введен в 1956 г., а в 1964 г. появилась первая версия системы DENDRAL (экспертной системы, предназначенной для определения состава сложных органических соединений), которую принято считать одной из первых ЭС. Интерес к системам ИИ во всем мире существенно возрос после того, как Япония объявила о проекте создания ЭВМ пятого поколения (1979 г.). Об этом прежде всего свидетельствует значительное увеличение ассигнований на разработку систем ИИ. Так, в период 1984-1988 гг. на эти цели в США было выделено 100 млн. дол., в странах ЕЭС - 90, в Японии - 30 млн. дол. Весьма симптоматично и то, что в последнее время на научных конференциях, связанных с работой сложных систем (будь то мощные ускорители элементарных частиц или системы автоматизированного проектирования) обязательно уделяется внимание экспертным системам. Это означает, что либо без таких систем в данной области уже нельзя обойтись, либо, что их использование признано экономически выгодным.

По функциональному назначению ЭС можно разделить на следующие типы. Во-первых, это мощные ЭС, рассчитанные на узкий круг пользователей (например, ЭС, предназначенные для управления технологическими процессами, или некоторые ЭС военного применения). Такие системы обычно работают в реальном масштабе времени и являются

довольно дорогостоящими. Во-вторых, это мощные ЭС, рассчитанные на широкий круг пользователей. К ним можно отнести системы медицинской диагностики. Базы знаний этих систем стоят недешево, так как должны содержать уникальные знания, полученные у экспертов высокой квалификации. Затраты на создание базы знаний могут составлять до 3 млн. дол. Сбором знаний и формированием баз знаний занимается специалист по сбору знаний (часто называемый инженером-когнитологом). Затраты на такого специалиста нолчас превышают 100 тыс. дол. в год.

Кроме систем названных типов, существуют и более простые ЭС, содержащие небольшое число правил (в виде которых чаще всего представляются знания в базе знаний) и, соответственно сравнительно недорогие. Это системы, рассчитанные на массового потребителя (например, системы, облегчающие поиск неисправностей в различной аппаратуре). Применение таких систем позволит организации сократить расходы на подготовку квалифицированного персонала и существенно уменьшить время поиска и устранения неисправностей. Базу знаний такой системы иногда можно заполнить, даже не прибегая к консультации квалифицированных экспертов, а используя сведения из справочных пособий и технической документации.

И наконец, есть довольно простые ЭС, предназначенные для индивидуального использования. Это может быть даже "самодельная" система, применяемая изготовителем, которому часто приходится разбираться в запутанных ситуациях. Если такому специалисту удалось выработать ряд эмпирических правил, то он может существенно облегчить себе повседневную работу, организовав эти правила в некоторую базу знаний и создав на ее основе свою экспертную систему. Системы такого типа находят применение в юриспруденции, в сфере коммерческой деятельности и т. п.

Растет интерес к созданию и использованию ЭС и в нашей стране, на что указывает ежегодно увеличивающееся число публикаций, посвященных этой тематике. В частности, несколько книг было выпущено издательством "Финансы и статистика". В предлагаемой вниманию читателей книге американских авторов Б.Сойера и Д.Фостера, в отличие от большинства книг на эту тему, изданных в нашей стране, описан практический подход к созданию ЭС. В книге схематично рассмотрены все основные стадии процесса

создания ЭС, начиная от постановки целей и кончая программированием. Кромс того, в ней описаны все основные подсистемы, входящие в состав "традиционной" ЭС. Здесь же приведены тексты программ различных модулей ЭС. Эти программы представляют собой готовую "оболочку" ЭС, которая, будучи наполнена знаниями, становится действующей экспертной системой. Читатель, лишь поверхностно знакомый с программированием, может взять из книги готовые тексты программ и построить свою собственную экспертную систему, наполнив ее базу знаний правилами, написанными на русском языке. Более искушенный читатель может на примере описанной системы разобраться в общих принципах построения программ такого рода и сконструировать собственную систему для решения конкретных задач. В любом случае книга заслуживает внимания широкого круга читателей, которых привлекает данная тематика, и несомненно вызовет большой интерес.

В.Н.Ивашков,
чл.-кор. АН СССР

* * *

Есть целый ряд людей, заслуживающих благодарность за помощь с их стороны. Я хотел бы поблагодарить Денниса Фостера, а также нашего редактора и литературных агентов. В полезных дискуссиях участвовали Том Шварц, Брюс Д'Амброзио и Элвин Шуп. Особо хотелось бы упомянуть Джона Пламмера (который доказал, что на самом деле существуют правила выбора галстука) и Чарльза Рубина. И наконец, я весьма признателен Рону Джеффрису за многолетнюю помощь и поддержку.

Бриан Сойер

Один мой знакомый композитор сравнивал гармонию с двумя различными мелодиями, которые вместе образуют более совершенное целое. Это сравнение во многом подходит к нашему союзу. Я хотел бы выразить свою признательность и благодарность Майку Ларсену и Элизабет Помада, литературным агентам высшего класса, и Терон Шрив из издательства John Wiley & Sons, Inc. А также моей замечательной жене Дон за ангельское терпение и моральную поддержку, и Мом, которая притворяется, что читает все, что я пишу.

Деннис Фостер

ПРЕДИСЛОВИЕ

О мудрых машинах-советчиках человечество мечтало с древнейших времен. Изречение рекомендаций к действию, основанных на мудрости богов, входило в обязанности греческих оракулов. Их современный аналог - компьютерная экспертная система - близок к воплощению этой древней мечты. Экспертные системы - подраздел несколько туманной отрасли информатики, именуемой искусственным интеллектом (ИИ), - по-видимому, "рассуждают" о реальном мире, подражая процессу мышления человека и используя логические взаимосвязи, которые подчас граничат с метафизическими. В отличие от простой обработки данных экспертная система оперирует символьными представлениями о действительности с помощью структуры, копирующей процесс анализа некоторой конкретной области экспертом - человеком.

В последнее время интерес к программированию экспертных систем чрезвычайно возрос, перешагнув за рамки чисто академических исследований и проникнув в область коммерции и другие сферы профессиональной деятельности. Сейчас экспертные системы используются практически во всех отраслях коммерции и управления: от финансовых служб и производства до систем военного назначения и образования. Первоначальный энтузиазм, которым характеризовалось взрывообразное развитие экспертных систем в середине 80-х годов, уступил место более реалистическому взгляду как на их возможности, так и на ограничения. Тем не менее, экспертные системы реально помогают решать насущные практические проблемы в деловой сфере, а влияние технологии ИИ и экспертных систем на жизнь человеческого общества в ближайшем будущем будет быстро расти.

В этой книге авторы рассматривают цели искусственного интеллекта, практические применения экспертных систем, архитектуру программы простой экспертной системы и приводят ряд методов программирования экспертных систем для использования на персональном компьютере.

Авторы предполагают, что читатель владеет элементарными навыками программирования на языке Ассемблера или языке высокого уровня. Читатели, уже умеющие манипулировать файлами и управлять базами данных, не будут испытывать затруднений при знакомстве с основными

алгоритмами рассматриваемых примеров. Книга предназначена для администраторов, инженеров по знаниям и пользователей, интересующихся как логической, так и архитектурной концепциями, лежащими в основе экспертной системы.

Предлагаемая вниманию читателя книга состоит из трех разделов. В первом разделе рассматривается развитие экспертной системы в перспективе: от ее корней в классической философии до современных коммерческих применений. Во втором - демонстрируется методика построения экспертной системы от элементарной инженерии знаний до архитектуры системы. Для читателя, желающего построить систему самостоятельно, приведен подробный текст программы на языке Паскаль, который с точки зрения переносимости программ на другие компьютеры и экономичности наиболее подходит для массовой аудитории. Третий раздел описывает экспертную систему в действии, на примерах решения некоторых обычных жизненных ситуаций, и завершается обзором способов организации взаимодействия экспертной системы с другими компьютерными программами.

Приведенные в настоящей книге программы написаны с помощью Турбо Паскаля фирмы Borland - популярного и недорогого компилятора для IBM PC и совместимых с ним компьютеров, требующего не менее 128 К оперативной памяти. Но эти программы могут использоваться не только для Турбо. Небольшие изменения позволяют применять их и для других популярных версий языка Паскаль, таких как Apple 1.1 или UCSD Pascal. Эти изменения приведены в приложении Б.

Некоторые версии Паскаля, в том числе и Стандартный Паскаль, не имеют встроенных операций над строками, необходимых программе для чтения и анализа содержимого файлов. В этом случае необходимо включить в программу тип "строка" и ряд функций для обработки строки, что требует более глубокого знания языка Паскаль.

Программы, приведенные в данной книге, предназначены главным образом для иллюстрации основных алгоритмов, составляющих инструментарий экспертной системы, а не для создания ее идеальной рабочей версии. В завершенном виде программа должна правильно реагировать на ошибки и оказывать разнообразную помощь пользователю, но соответствующая доработка привела бы к существенному увеличению объема программы и затруднила бы восприятие ее структуры.

ПРОЛОГ

СИТУАЦИЯ. Однажды, прогуливаясь в пригородной роще, вы обнаруживаете только что приземлившийся корабль пришельцев. Природная любознательность побуждает вас войти внутрь. Появившиеся инопланетяне, не проявив ни малейшего дружелюбия, молча приводят вас в пустую каюту и там оставляют.

ВАША ЦЕЛЬ. Освобождение.

Вы успели заметить, что корабль представляет собой многоэтажную громадину со сложным переплетением отсеков, коридоров и прочих помещений. От входа, расположенного на первом этаже, вас заставили повернуть направо и в полной темноте вы прошли около 50 метров, затем сделали поворот на 45°. Насчитав 52 шага, вы вошли в лифт, который, судя по инерции, поднялся наверх. Когда створки лифта распахнулись, захватчики провели вас на 48 шагов вперед, а затем повернули на 45° влево. Через 18 шагов был еще один поворот на 90° влево. Отсчитав восемнадцать шагов, вы вновь повернули налево на 45°, и примерно через 50 шагов подошли к лестнице. Конвоиры провели вас на два пролета вверх и вывели в коридор. Вы сделали 48 шагов прямо и повернули налево примерно на 45°. Еще 12 шагов и наконец справа распахнулась дверь помещения, в котором вас и оставили.

В этой комнате стены выкрашены в черный цвет. Сквозь крошечный иллюминатор в вашу темницу пробивается слабый свет. По мере того как томительно тянутся дни, вы замечаете, что прямые лучи солнца в комнату не попадают. На пятый день заточения сопровождающие выводят вас из комнаты и отводят в лабораторию, уставленную медицинским оборудованием, где вас обследуют. Проходя по коридору, вы замечаете, что все вокруг серого цвета: металлические стены, двери, потолок и даже пол. В лифте вы поднимаетесь наверх. Здесь стены тоже выкрашены серым цветом. По пути в лабораторию вы сделали три поворота: на 45°, на 90° и еще раз

на 45°. Лаборатория находится в конце коридора. Сквозь ее иллюминаторы проходят прямые солнечные лучи, образуя овальные пятна на сером металлическом полу.

Спустя три дня за вами снова приходит молчаливая охрана. Лифт поднимает вас на два этажа вверх (судя по времени подъема). Стены коридора выкрашены в зеленый цвет. Охранник отводит вас на 8 шагов от лифта и поворачивает направо. Вы оказываетесь в комнате с глухими, без иллюминаторов, стенами.

Теперь у вас есть вся информация, необходимая для того, чтобы проложить маршрут к выходу.

Ваша задача: построить набор правил для разработки маршрута к освобождению из любой точки корабля.

БАЗА ЗНАНИЙ

Все знания о расположении помещений внутри корабля, полученные вами на основе наблюдений, могут быть представлены в виде выражений, описывающих факты, и эвристических правил. Некоторые события, очевидцем которых вы были, можно выразить так:

Прямые лучи солнца не попадают в иллюминатор.
Прямые лучи солнца попадают в иллюминатор.
Стены комнаты черного цвета.
Стены коридора зеленого цвета.
Стены коридора серого цвета.

А вот некоторые из правил, которые вы могли бы включить в *базу знаний* освобождения:

Если прямые лучи солнца не попадают в иллюминатор, то это северная сторона.
Если прямые лучи солнца попадают в иллюминатор, то это южная сторона.
Если стены коридора серого цвета, а направление южное, то место - лаборатория.
Если место - лаборатория, то этаж - первый.

С помощью этих и аналогичных выражений можно прийти к выводу, что ваша комната расположена в северной части корабля, и что шахта лифта находится слева от нее че-

рез два правых поворота в южном конце. Кроме того, можно заключить, что комната, в которую вас привели, находится на одном этаже с люком, расположенным в конце восточной части корабля, а пульт управления - в начале западной части. Вся совокупность эмпирических наблюдений о корабле составляет *экспертную систему*. Цель этой системы - прогнозирование на основе установленных или вероятных отношений для определения пути к выходу из любой точки корабля. Правдоподобность полученных решений будет зависеть от степени определенности фактов и правил, в соответствии с которыми они принимаются.

Например, если вы на 80% уверены в том, что "если стены = черные, то место = комната" и на 50% - "если место = комната, то этаж = второй", то уверенность, что вы находитесь на втором этаже, при условии, что стены имеют черный цвет может быть оценена в 90%. Существует множество способов подсчета неопределенности, от традиционной теории вероятностей до сугубо субъективных оценок.

Простая экспертная система, основанная на правилах, могла бы предоставить вам базу знаний в виде дерева решений, иерархию отношений "если - то", выводящих вас из любой точки лабиринта корабля к шахте лифта и вверх или вниз на необходимое количество этажей к выходу.

Если бы вы имели возможность запустить такую экспертную систему на компьютере, диалог мог бы выглядеть примерно так:

Какова цель консультации?
Определить путь к выходу.

Какого цвета стены в помещении?
1. серые
2. зеленые
3. черные
2

Какого цвета стены в коридоре?
1. серые
2. зеленые
2

Какой свет проходит через иллюминатор?
1. прямой солнечный
2. непрямой солнечный
3. нет иллюминатора
3

Я рекомендую выйти за дверь и повернуть направо. Пройдите примерно 48 шагов и поверните еще раз направо на 90°.
Когда вы дойдете до лифта, поднимитесь на один этаж и вылезайте через вентиляционный люк.

Подобно плану выхода из корабля, экспертная система "прокладывает" путь, используя причинно-следственные отношения для построения заключений на основе фактов и событий. Оценивая прошлое, она делает выводы относительно будущего.

ВВЕДЕНИЕ

Когда специалист по интеграции систем фирмы Digital Equipment Corporation приступает к выбору конфигурации компьютерной системы VAX-11 для конкретного пользователя, то первое, что он делает - это готовит себе кофе. Потом звонит своему эксперту по конфигурации и пока он пьет кофе, советчик оценивает спецификации пользователя, определяет оптимальный набор компонент и строит точную рабочую схему интеграции системы. Этот эксперт по интеграции никогда не видел настоящих составляющих VAX-11 и никогда не был в помещениях, где производится установка компьютерного оборудования. Этот эксперт - XCON - программа, разработанная объединенной группой специалистов фирмы DEC и ученых университета Карнеги-Меллона для определения конфигурации сложных компьютерных систем.

Сталкиваясь с загадочным набором симптомов, который указывает на заболевание сердечного клапана, практикующий врач из Огайо обращается за частной консультацией к своему весьма уважаемому коллеге. Детально рассмотрев историю болезни пациента, симптомы заболевания и результаты лабораторных исследований, последний быстро ставит наиболее вероятный диагноз: недостаточность трехстворчатого клапана. Как ни странно, этот выдающийся коллега никогда не получал диплома о медицинском образовании и не провел ни единой минуты у постели больного. Этот коллега - компьютерная экспертная система, специализирующаяся на медицинских диагнозах.

Благодаря своему персональному компьютеру, метеоролог из Национальной Службы погоды предупреждает о возможном наводнении в Аризоне. В течение нескольких минут он с восхищением наблюдает, как экспертная система планирует прогноз и формирует комплекс предупредительных мероприятий.

Специалист по бурению скважин на юге Франции,

конструктор электронного оборудования в Японии, логопед в Юте - всех их объединяет знакомство с этими профессионалами, которые могут быть советчиками, конструкторами, диагностами. Это исполнительные помощники при принятии решений, наставники, исследователи. Они приходят к своим выводам на основе наблюдений, сделанных человеком, они "рассуждают" об окружающем мире в терминах, придуманных человеком, и используют человеческую логику для решения проблем, относящихся к человеку. Но эти знакомые - компьютерные экспертные системы, являются чем угодно, *только не* человеком.

Однако подобно эксперту-человеку экспертная система решает проблемы, которые подчас определены лишь в общих чертах. Она "рассуждает" о проблеме, используя логические взаимосвязи аналогично тому, как это делает человек. Обычно такой процесс включает в себя: (1) определение, (2) исследование, (3) принятие гипотез, (4) решение и (5) общение.

▩ *Определение* требует анализа проблемы в плане причинно-следственных отношений, а не просто в связи с сопутствующими ей эмпирическими событиями. Характерной чертой экспертной системы является то, что она всегда "предвидит" дальнейшие события.

▩ *Исследование* заключается в получении ответов на вопросы о конкретных деталях каждого события, его характеристиках, связях и соотношениях. Часто экспертная система "объясняет" метод своих рассуждений.

▩ *Принятие гипотез* связано с формированием пробных предположений и проверкой их достоверности с помощью алгоритмов, основывающихся на имеющейся информации и анализе возможных результатов. Некоторые экспертные системы формируют параллельные гипотезы, используя их конкуренцию для получения оптимального решения.

▩ *Решения* - это соответствующим образом сформулированные рекомендации к действиям, направленным на достижение конкретных целей. Экспертные системы не просто представляют алгоритмы, они предлагают конкретный совет.

▩ *Общение* предполагает распознавание информации, получаемой от человека в произвольной форме, и представление рекомендаций в понятном и пригодном к практическому использованию виде. Экспертная система "разговаривает" на естественном языке, а не на языке операционной системы компьютера.

Нетрудно видеть, что для создания экспертной системы нужны специалисты из нескольких различных областей: прежде всего, эксперт в предложенной предметной области; во-вторых, инженер по знаниям, который может разложить процесс проведения экспертизы на символьные и логические составляющие; и, в-третьих, программист для создания рабочей среды. На практике, для разработки технологии часто собираются большие коллективы специалистов в каждом из направлений.

Кроме того, очевидно, что создание экспертных систем в принципе отличается от написания компьютерных программ других типов. Давайте на минутку вернемся к экспертной системе, которую мы сконструировали в прологе. Хотя предложенная там ситуация довольно тривиальна, она хорошо иллюстрирует основные отличия экспертных систем от обычных программ обработки данных. Отметим следующие основные моменты.

Во-первых, традиционные компьютерные программы детерминированы; при решении любой поставленной задачи они используют одну и ту же последовательность операций. Экспертная система строит собственное дерево решений для достижения каждой новой цели.

Во-вторых, в отличие от традиционной программы, которая строится главным образом из линейных отношений, экспертная система обрабатывает произвольные *символьные* выражения (например, концептуальные, временные и пространственные отношения). Цель обычной компьютерной программы - расчет числовых значений, накопление констант и извлечение данных из памяти. Цель экспертной системы состоит в выдаче рекомендаций, основанных на предсказываемом поведении наблюдаемых объектов и течении событий. Кроме того, традиционные программы имеют дело с общепризнанными фактами: например, никто не станет обсуждать значение логарифма или произведения 10 на 10. В противоположность этому экспертная система заостряет внимание на эмпирических ассоциациях, которые описывают общие понятия и события. Такая информационная база образуется путем обработки знаний, накопленных экспертами-людьми в конкретной области. В этом отношении специализированная экспертиза более жизненна, чем точные науки.

В-третьих, сконцентрированная экспертиза выражается

в виде наборов описательных атрибутов и неожиданных взаимосвязей. Поэтому если традиционная компьютерная программа следует определенным математическим правилам, то работа экспертной системы строится на обработке символьных выражений, основанной на эвристических рассуждениях.

Вследствие субъективного характера базы знаний, уверенность в некотором факте или правиле может быть не абсолютной. Вот почему экспертные системы часто основаны на выражениях относительной уверенности. Рассмотрим, например, такое предложение: "Если животное - птица, то оно умеет летать". Здесь уверенность основана на следующем факте: "Птица умеет летать". А если эта птица - индейка, страус или, скажем, пингвин?

Эксперт по птицам мог бы придать этому правилу некоторый коэффициент уверенности; допустим, что ему кажется весьма правдоподобным, что если животное - птица, то оно умеет летать. В этом случае правило может быть записано в следующем виде:

```
если
животное=птица,
то
умеет=летать, ку=90
```

Целая величина *ку* в данном выражении обозначает коэффициент уверенности, равный 90.

Нетрудно заметить, что определение уверенности экспертом является весьма существенным в таких утверждениях, как: "Если боль в груди отдает в левую руку, то у больного инфаркт миокарда" или "Если цена перепродажи заранее оговорена в контракте, то поставщик неверно устанавливает цену".

Экспертная система имитирует рассуждения человека, выдавая предполагаемые решения определенной проблемы, а затем выделяя наиболее подходящее из них. Это позволяет ей с самого начала отбросить бесполезные решения. Более того, она использует составную структуру независимо приобретенных субъективных знаний, применяя разработанную человеком систему проведения экспертизы к решению жизненных проблем. Благодаря системному анализу проблемы с различных точек зрения, она выдает не просто подходящее, а наилучшее решение.

РАЗДЕЛ 1

ЭКСПЕРТНЫЕ СИСТЕМЫ В ПЕРСПЕКТИВЕ

ГЛАВА 1

Искусственный интеллект и экспертные системы

Хотя мы часто считаем экспертные системы новой технологией, концепции их построения стары, как род человеческий. Ученые древности попытались систематизировать все накопленные человечеством знания, а затем начался 30 000-летний поиск разрешения загадки познания. Еще за 700 лет до н.э. ученые Вавилонии разработали сложную совокупность правил вида "если - то", описывающих эмпирические взаимосвязи между наблюдаемыми явлениями повседневной жизни. Многие из них представляли собой ранние попытки разработать систему медицинской диагностики. Например, одно из правил гласило:

> Если человек холоден и горяч
> одновременно... он умрет неожиданно.

Субъективность эмпиризма была излюбленной темой Декарта, который верил, что эмпирическое знание может быть выведено с той же степенью определенности, что и математические теоремы. Его методологический скептицизм, отбрасывание гипотез посредством последовательной проверки, составляет парадигму системы исследования, используемой современными экспертными системами.

Однако наибольшее влияние на современных программистов экспертных систем оказал Джон Дэйви. Он рассматривал логику не как академическое упражнение, а как конкретный метод исследования, применимый к жизненным проблемам.

Дэйви писал: "Процесс получения представления о том, что отсутствует на основе того, чем мы владеем есть вывод. Каждый вывод, именно потому что он выходит за рамки установленных и известных фактов, которые получены либо из наблюдений, либо из собранных ранее знаний, влечет за собой скачок от известного к неизвестному". Далее Дэйви подчеркивал необходимость подтверждения вывода и проведения различий между эмпирическими и предсказанными причинно-следственными связями. "Весьма существенно,–утверждает он, - чтобы каждый вывод был проверенным выводом".

Этими тремя предложениями он, по существу, описал современную, основанную на правилах экспертную систему.

Хотя машины, имитирующие рассуждения человека, относительно новы, этого нельзя сказать о самой идее создания механического устройства, выносящего суждения, подобные суждениям человека. В конце 30-х годов, когда инженеры начали разрабатывать планы первых электронных "думающих машин", встал вопрос о том, должно ли такое устройство "считать" или "рассуждать". Британский ученый Алан Тьюринг утверждал, что такая машина должна не просто обрабатывать числовые величины, а моделировать процесс рассуждения. Однако в то время военные применения были более насущными и идеи Тьюринга остались без внимания.

Спустя 10 лет, когда падение цен на средства вычислительной техники сделало их более широкодоступными, различные исследовательские группы как в академической, так и в коммерческой областях вновь стали проявлять интерес к гипотезам Тьюринга. В 1956 г. логики Массачусетского Технологического института и Станфордского университета, философствующие о методах механического повторения процесса познания, предложили термин "искусственный интеллект" (ИИ). Но до 60-х годов идея создания машины, которая может "рассуждать", оставалась чисто теоретической.

Важная основополагающая концептуальная работа была выполнена Алланом Ньюэллом и Гербертом Саймоном (университет Карнеги-Меллона), которые предложили схему процессов, лежащих в основе решения проблем человеком. Предложенная ими программа называлась универсальным решателем задач (GPS - General Problem Solving), и была первой ИИ, использующей нечто напоминающее правила. Наиболее ранней попыткой разработки практического применения была,

по-видимому, DENDRAL - программа расшифровки масс-спектрограмм, созданная в Станфордском университете в середине 60-х годов для выделения органических соединений. К концу десятилетия ряд коллективов-разработчиков из частного сектора начал выпускать автоматизированные системы, основанные на символьной обработке и предназначенные для замены специалистов-людей. Применяя большие базы накопленных знаний и общение на языке, близком к естественному, эти компьютерные программы-советчики продемонстрировали широкие возможности символьной обработки, построенной на правилах. Кроме того, их развитие заложило хорошую основу для исследований в области природы обучения и архитектуры познания, знаний, необходимых для создания машинных систем логического вывода. В 1971 г. компания AML International, специализирующаяся на автоматизации медицинского оборудования, выпустила первую компьютерную медицинскую диагностическую систему. Эта программа не только содержала обширную базу знаний в области медицинской экспертизы, но и непосредственно взаимодействовала с диагностическим оборудованием, таким, как измерители спирометрии, кровяного давления и оптометрических характеристик.

Но более известный предшественник современных, основанных на правилах, экспертных систем был создан в 1974 г. Станфордской группой HHP (Heuristic Programming Project). Это - программа MYCIN, при составлении которой использовались многие из теоретических академических разработок в таких областях как символьная математика, машинная эвристика и инженерия знаний. Проект включал две задачи: систематизированный диагноз бактериальных инфекций и рекомендации по курсу эффективной терапии. Ядро MYCIN составляли: исчерпывающая база знаний, простой, основанный на правилах механизм обработки, и символьная математика. Другими словами, она представляла собой работающую модель фактически всех используемых в настоящее время экспертных систем и предшественницу всех своих последующих конструктивных аналогов. Возможности MYCIN несколько ограничивались тем, что она полностью зависела от ввода клинических проявлений оператором и была неспособна пополнять свою базу данных путем статистического обобщения новых результатов. Несмотря на свои недостатки этот эксперимент получил широкое признание, как важная веха в исследованиях ИИ, и в литературе по

экспертным системам его описание занимает значительное место.

В 80-х годах исследования ИИ стали почти синонимом исследований в области военной технологии, более известных как программа расширенных исследований в области обороны (DARPA). Одним из результатов таких исследований было появление облегченного алюминиевого танка-робота, способного распознавать рельеф местности, самостоятельно маневрировать и стрелять без управления по заданным целям. Характеристики танка в приближенных к боевым условиях оказались довольно скромными, но сам факт, что он смог без человеческого вмешательства выполнить свою ограниченную задачу, свидетельствует о значительном продвижении в практическом применении теории ИИ.

В 1982 г. Министерство внешней торговли и промышленности Японии усилило интерес к исследованиям в области ИИ, объявив о проекте создания ЭВМ пятого поколения. При разработке четырех первых поколений компьютеров использовалась исключительно архитектура фон Неймана, традиционная конструкция компьютеров, включающая в себя арифметико-логическое устройство и несколько перезагружаемых регистров. В противоположность этому цель японской группы заключалась в создании пятого поколения компьютеров, познавательные способности которых будут неотличимы от человеческих. Основное внимание в проекте уделялось экспертным системам, программам естественного языка, распознавания голоса и изображений, робототехнике и суперкомпьютерам.

В этом исследования экспертных систем расходятся с целями проектов ИИ. Энтузиасты ИИ часто рассуждают об автономных машинных сущностях, которые будут взаимодействовать с человеческим познанием, но не основываться на нем. В противоположность этому *экспертные системы всецело зависят от человеческой экспертизы.* Хорошо разработанная экспертная система способна выводить суждения с такой же (или почти с такой же) точностью, как и эксперт-человек именно потому, что ее знания основаны на человеческом опыте. Эффективность такой системы фактически определяется базой знаний, созданной человеком.

Полезность экспертных систем была продемонстрирована в ряде приложений, включая минералогию, интеграцию компьютерных систем и военные

интеллектуальные системы. Одна из таких программ - Prospector - представляет уникальный "сплав" академической науки, общественных служб, правительства и личной инициативы. Предназначенная для управления геологическими исследованиями месторождений минералов, эта система была разработана в международном центре SRI International под эгидой Национального научно-исследовательского центра Геологического управления США. Она прекрасно зарекомендовала себя на испытаниях, но никогда не применялась на коммерческом уровне. Однако заложенные в ней концепции позднее были использованы в проекте Drilling Adviser - микрокомпьютерной экспертной системы, помогающей разрешать проблемы, возникающие при полевом бурении нефтяных скважин.

В течение почти десяти лет эта мощная технология не выходила за рамки стен университетских и военных лабораторий. В настоящее время интерес к практическим экспертным системам, питаемый значительными финансовыми инвестициями из частного сектора, достиг максимума. В результате колоссально возросло число программных продуктов, предназначенных для создания систем, основанных на знаниях, с диапазоном применений от дегустации вин до оценки служащих. Однако лишь малая часть этих программ, по-видимому, оставит след в истории экспертных систем. Стоимость этих инструментов составляет от нескольких сот до нескольких тысяч долларов. Кроме того, они имеют ряд общих характеристик. Все они основываются на базах знаний, полученных из опыта экспертов, используют символьные выражения фактов и соответствующих атрибутов, снабжены эвристической машиной вывода (а большинство работают с недоопределенными знаниями), решают проблемы, опираясь на человеческий опыт.

ГЛАВА 2

Элементы экспертных систем

Все экспертные системы включают в себя по крайней мере три основных элемента: базу знаний, машину вывода и интерфейс пользователя. *База знаний* содержит информацию о том, что известно о данном предмете в настоящий момент. *Машина вывода* обеспечивает применение того, что известно к тому, что еще не известно. Интерфейс пользователя способствует взаимодействию между системой и пользователем. Взятая как целое, экспертная система моделирует знания эксперта и умение их применять.

Общее представление о функционировании экспертной системы дает рис. 2.1 . Группа экспертов или иной источник экспертизы обеспечивает загрузку в базу знаний фактов, наблюдений и способов анализа ситуаций. Пользователь запрашивает систему о конкретных проблемах через интерфейс, который допускает общение с использованием не технических, а обычных выражений. В мощных интеллектуальных системах существует *интерфейс на естественном языке*, который позволяет задавать вопросы и получать ответы на обычном английском или русском языках. В менее мощных системах пользователю предоставляется не столь изысканный, но тем не менее "дружественный" интерфейс. Информация, содержащаяся в базе знаний, обрабатывается с помощью машины вывода, которая использует эмпирические ассоциации - или правила "если-то" - для формирования и проверки возможных решений. Интерфейс пользователя в доступной форме передает полученные результаты оператору.

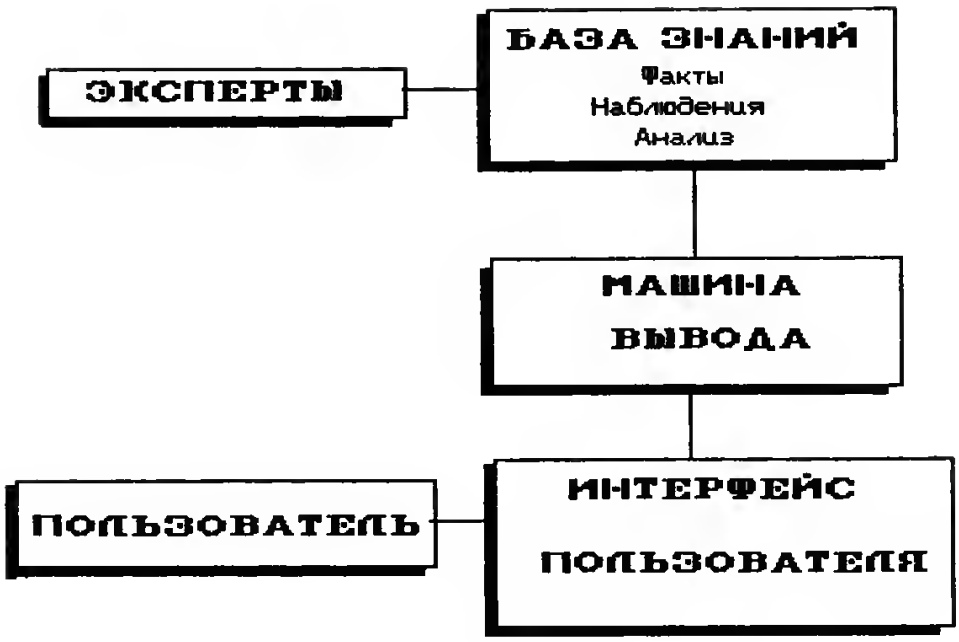


Рис. 2.1. Элементы экспертной системы

База знаний

База знаний содержит известные факты, выраженные в виде объектов, атрибутов и условий. Помимо описательных представлений о действительности, она включает выражения неопределенности - ограничения на достоверность факта.

В этом отношении она отличается от традиционной базы данных вследствие своего символьного, а не числового или буквенного содержания. При обработке информации базы данных, пользуются заранее определенными логическими правилами. Соответственно, база знаний, представляющая более высокий уровень абстракции, имеет дело с *классами* объектов, а не с самими объектами.

База знаний создается людьми - консультантами, авторами учебников, исследователями - либо самими экспертами, либо на основе их работ. В поисках источника для наполнения базы знаний жизненный опыт важнее, чем высокий интеллект. Эксперт, который исходит из продолжительных наблюдений за событиями в некоторой конкретной области, скорее всего, создаст более полезную базу знаний, чем гениальный

аналитик, который больше полагается на интуитивное проникновение в сущность явлений. Предположим, например, что вы строите экспертную систему выбора компьютера для различных применений. Блестящий инженер-конструктор, отлично владеющий методами разработки интерфейсов, на практике будет менее полезен, чем консультант по применениям, который наблюдал большое количество установленных систем, выполняющих предназначенные для них задачи.

Машина вывода

Главным в экспертной системе является механизм, осуществляющий поиск в базе знаний по правилам рациональной логики для получения решений. Эта машина вывода приводится в действие при получении запроса пользователя и выполняет следующие задачи:

сравнивает информацию, содержащуюся в запросе пользователя, с информацией базы знаний;

ищет определенные цели или причинные связи;

оценивает относительную определенность фактов, основываясь на соответствующих коэффициентах доверия, связаных с каждым фактом.

Как следует из ее названия, машина вывода предназначена для построения заключений. Ее действие аналогично рассуждениям эксперта-человека, который оценивает проблему и предлагает гипотетические решения. В поиске целей на основе предложенных правил, машина вывода обращается к базе знаний до тех пор пока не найдет вероятный путь к получению приемлемого результата. Например, программа медицинской диагностики сначала пытается выделить специфический болезнетворный организм, анализируя список на первый взгляд не связанных симптомов, а затем определяет курс эффективной терапии. Другой пример - программа оптимизации прибылей. Она оценивает различные воздействия на поток платежей для определения последовательности действий, которые способствуют увеличению поступлений при уменьшении затрат и налогов.

Интерфейс пользователя

Задача интерфейса пользователя состоит в организации обмена информацией между оператором и машиной вывода. Интерфейс с использованием естественного языка создает видимость произвольной беседы, применяя повседневные выражения в правильно построенных предложениях. Очевидно, что чем более естествен такой интерфейс, тем выше требования к внешней и оперативной памяти. Следовательно, системы, предоставляющие пользователю максимум удобств, расходуют больше ресурсов основной машины, доступных с удаленных рабочих станций. Инструментальные средства, предназначенные для работы на персональных компьютерах, имеющих ограниченные возможности, неизбежно приносят "дружественность" в жертву эффективности.

Интерфейс *прямого ввода* должен уметь распознавать язык, или, по крайней мере, достаточное количество ключевых слов и фраз, чтобы улавливать их связь с рассматриваемой проблемой и ее предполагаемыми решениями.

Допустим, что вы - владелец шикарного французского ресторана и хотите создать у себя превосходный винный погреб. Для того чтобы узнать, какие вина и в каком количестве вам закупать, вы решили проконсультироваться с экспертной системой. Ваша консультация могла бы выглядеть примерно так:

Какую из ведущих виноделъческих провинций вы предпочитаете?
Бургундию.

Сколько в среднем бутылок вина этого типа вы приобретаете в месяц?
60

Через сколько месяцев ваши запасы истощаются и вы повторяете закупку?
6 месяцев

В этом примере, интерфейс должен понять слово "Бургундия" и знать относительное значение каждой введенной числовой величины.

Система непрямого ввода менее специфична, но улучшает эффективность интерфейса. Вот как тот же самый сценарий мог бы выглядеть при интерфейсе непрямого ввода:

Какую из ведущих винодельческих провинций вы предпочитаете?
1. Бургундия
2. Бордо
1

Сколько в среднем бутылок вина этого типа вы приобретаете в месяц?
1. меньше 20
2. между 20 и 40
3. между 40 и 60
4. свыше 60
3

Через сколько месяцев ваши запасы истощаются и вы повторяете закупку?
1. 3 месяца или меньше
2. между 3 и 6 месяцами
3. между 6 и 12 месяцами
4. свыше 12 месяцев
3

Более мощные экспертные системы объясняют, почему задан тот или иной вопрос и как был сделан соответствующий вывод. Эти объяснения получаются на основе эвристических правил, с помощью которых происходит управление взаимодействием машины вывода с базой знаний. В нашем примере эксперт по винам мог бы объяснить свои вопросы следующим образом:

Какую из ведущих винодельческих провинций вы предпочитаете?
почему

Ваш ответ на этот вопрос поможет мне определить, применимы ли следующие правила:

если провинция =Бордо,
то покупайте у агента;
если провинция =Бургундия,
то покупайте у оптовика;

Это изречение основано на том факте, что в Бургундии (винодельческой провинции Франции) виноградники разделяются по поколениям. В результате у одного агента можно приобрести вина различных сортов, в то время как в Бордо агенты обеспечивают покупателя вином одного сорта.

Интерфейс пользователя подчас служит определяющей мерой достоинств экспертной системы, когда простота общения играет не меньшую роль, чем эффективность машины вывода или полнота базы знаний.

Человеческий аспект

В работе с экспертными системами принимают участие как минимум три группы людей. Во-первых, *администрация* устанавливает предназначение экспертной системы, ограничивает предметную область, которую должна охватывать система, и точно определяет, какие выгоды организация сможет извлечь из ее применения. Во-вторых, специалист по сбору знаний *(инженер-когнитолог)* собирает информацию, необходимую для базы знаний, сравнивает соответствующие данные и эвристически организует информацию. В-третьих, потенциальный *пользователь* указывает, как будет использоваться система, какого рода проблемы предстоит решать и каким образом будет осуществляться взаимодействие программы с оператором. И наконец, системе требуется *эксперт* (чаще группа экспертов) в установленной предметной области, для получения от него знаний, как в форме фактической информации, так и относительно аналитических методов, которые применяются для решения проблем в этой области. Человеческий аспект экспертных систем представлен на рис. 2.2.

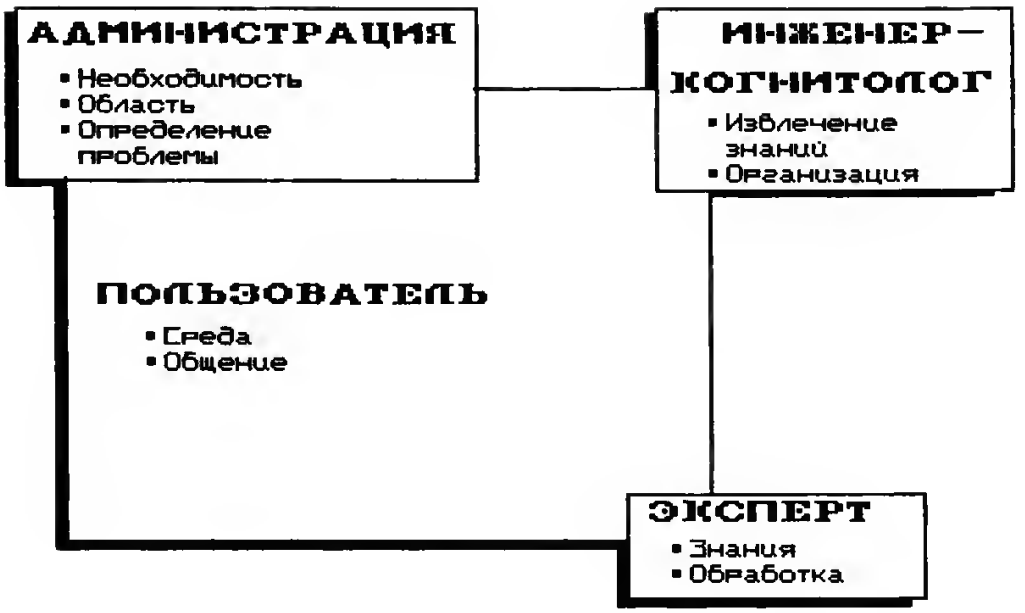


Рис. 2.2. Человеческий аспект экспертной системы

Машинный аспект

Компьютерную часть системы представляют компоненты программного обеспечения, которые обрабатывают полученную информацию о действительности, заложенную в символьном виде в базу знаний. Все экспертные системы имеют по крайней мере три таких компонента: базу знаний, машину вывода и интерфейс пользователя.

В базу знаний поступают факты. Связь между фактами представлена эвристическими правилами - выражениями декларативного знания об отношениях между объектами. Каждое такое правило имеет составляющую "если" (предпосылку) и компонент "то" (заключение), которые определяют прямую или обратную причинно-следственную связь. Рассмотрим следующее правило: "Если у пациента почечная инфекция, то у него наблюдается повышенная температура и боли в нижней части спины". Это выражение определяет прямую связь для предсказания результата. Следующее правило, наоборот, образует обратную цепочку для предположения о причине: "Если у пациента жар и он жалуется

на боли в нижней части спины, то можно предположить, что это почечная инфекция".

Действительные утверждения только вероятны, другими словами, степень их определенности не всегда абсолютна. В отношении себя кто-то может со стопроцентной определенностью утверждать: "Мой любимый цвет - красный". Но в большинстве утверждений уровень доверия может быть более или менее условным: "Если у человека красное лицо, то он вне себя от бешенства".

Следовательно, количественные коэффициенты определенности увеличивают точность рассуждения экспертных систем. Такие выражения относительной уверенности часто основываются на статистических, вероятностных, или просто субъективных предпосылках. Общепринятая схема состоит в том, чтобы варьировать уровень доверия от 0, представляющего минимальную степень определенности, до 100 - высшей степени.

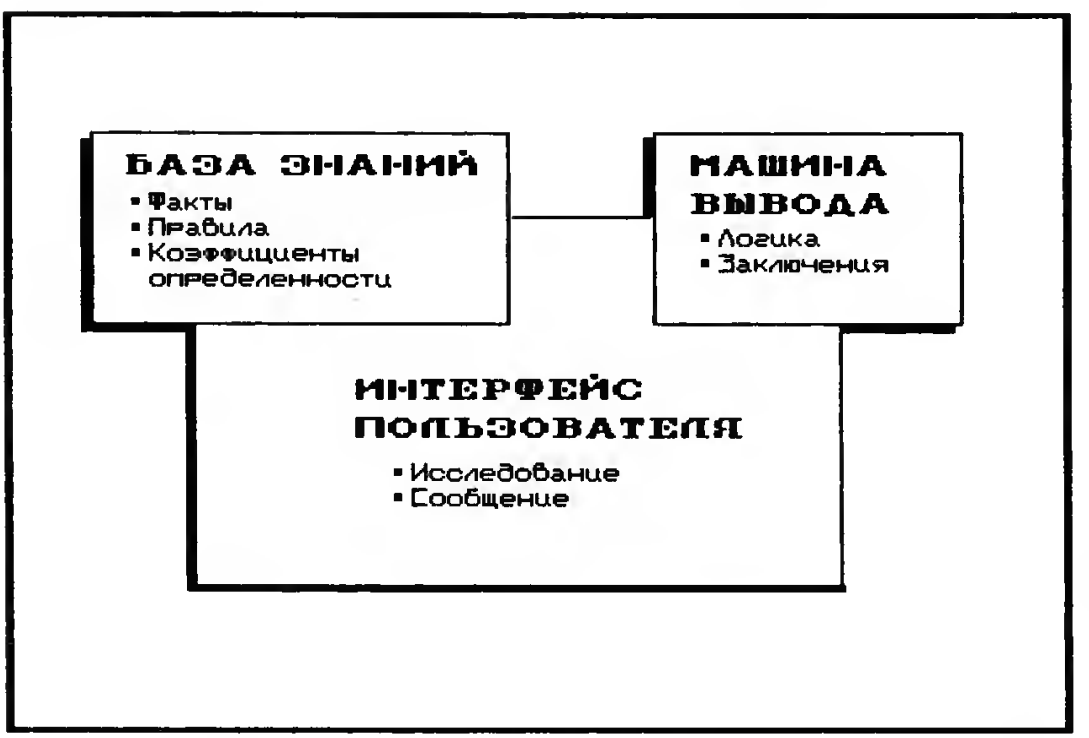


Рис. 2.3. Машинная часть экспертной системы

Машина вывода применяет логику, четкие рассуждения, установленные правилами, для проверки заключений и получения выводов. Через интерфейс пользователя эта система проводит свое исследование и поддерживает связь с человеком, который ею управляет. Рис. 2.3 суммирует машинную часть экспертной системы.

Взаимодействие "человек - машина"

Прежде чем приступить к рассмотрению этих компонентов в контексте программирования экспертных систем, выясним, как взаимодействуют человеческий и машинный аспекты в процессе создания базы знаний.

ГЛАВА 3

Планирование экспертной системы

Наука создания экспертных систем, как и вообще ИИ, большей частью экспериментальна. Таковой ее делает "скачок от известного к неизвестному". *Каждую экспертную систему следует рассматривать как эксперимент*, требующий подтверждения на основе эмпирических результатов.

Как и в любом эксперименте, возможность получения удовлетворительных результатов зависит от определения целей и точности постановки задачи.

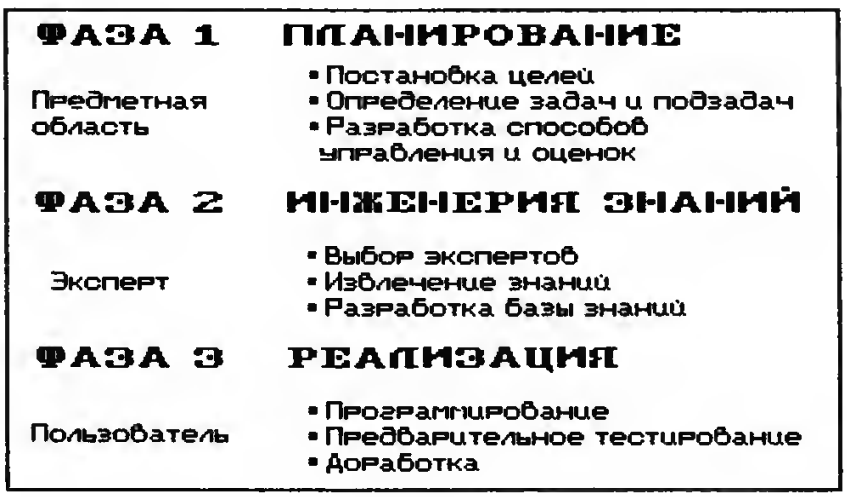


Рис. 3.1. Три фазы разработки экспертной системы

Кроме того, на развитие экспертных систем оказывают влияние три характерных фактора:

1. Предметная область.
2. Эксперт.
3. Пользовательская среда.

Взаимодействие этих факторов на протяжении трех ос-

новных фаз разработки проиллюстрировано на рис. 3.1. Процесс начинается с установления целей и определения специфической проблемы, которую предстоит решать экспертной системе.

Постановка целей

Любой эксперимент начинается с анализа целей проекта. Разработчики должны знать природу проблем, которые им предстоит решать. Они должны иметь возможность описать задачу, последствия или результаты, ожидаемые от ее решения, и значение этих результатов для их организации. Короче говоря, они должны точно знать, какую пользу принесет применение данной экспертной системы. Это нужно для того, чтобы устранить неопределенность и облегчить разработку специальных мер, способных увеличить достоверность результатов, расширить систему и сделать ее более определенной. Поставив правильные цели, планировщики программ получают хорошую возможность выбора точных задач и, следовательно, получения полезных решений.

Цели системы можно разделить на три типа: конечные, промежуточные и вспомогательные. *Конечная* цель описывает, какое действие, исход или результат должны получаться как следствие консультации. *Промежуточные* цели системы подразделяют общую проблему на подзадачи, описывая проблемы, которые должны быть решены для достижения конечной цели. *Вспомогательные* цели помогают планировщикам определить конкретные области экспертизы, требующиеся для решения перечисленных задач.

Каждая цель - это предполагаемый результат, который может быть получен, если программа предлагает решения специфических проблем. Установление их в качестве отправных точек нужно для того, чтобы дать разработчикам ясное представление, какие задания должна выполнять система и как она должна действовать при их выполнении.

Подробно мы рассмотрим эти цели в гл. 13.

Определение задач

Если цели определены, то типы проблем, которые предстоит решать и способ подхода программы к их решению становятся более очевидными.

Кроме того, в этой фазе велика потребность в ограничениях. Хотя база знаний должна быть детально исчерпывающей, но тем не менее ее следует ограничивать по фактам и правилам, требующимся для достижения поставленных целей. Например, возраст и вес больного с заболеванием сердца важны для назначения больничного режима, а цвет его глаз и острота слуха - нет.

При разработке инструментария экспертной системы желательно предусмотреть ее реальную необходимость для организации. Стоят ли предполагаемые результаты, чтобы тратить на них время, средства и усилия персонала? Например, вряд ли имеет смысл разрабатывать систему стоимостью в 100000 $ для уменьшения числа работающих по найму, если общие затраты на них составляют 10000 $ в год. Короче говоря, увеличится ли доход организации, если стоимость разработки экспертной системы вычесть из прибылей? Если ценность инструмента не превышает величины вложений, то его разработка неоправдана.

Сбор знаний

Вторая фаза включает в себя сбор, структуризацию и перевод основного содержимого экспертизы, требующегося для решения поставленных задач. В этом процессе обычно участвуют как специалисты в предметной области, так и инженеры-когнитологи. Тип и число экспертов часто определяются сложностью проблемы: может ли один эксперт, консультант или управляющий предоставить все необходимые знания? Или требуется группа экспертов? Небольшие базы знаний часто извлекаются из справочников или учебных пособий. Экспертные системы, создаваемые на персональных компьютерах, чаще всего ограничиваются небольшим кругом проблем, и базу знаний можно накопить самостоятельно, воспользовавшись помощью небольшой группы специалистов, или прибегнув к помощи справочных пособий. Содержание такой базы знаний обуславливается определением проблемы.

Это значит, что *база знаний может охватывать любую предметную область, которая описывается набором правил "если - то".* Она может быть проста как набор инструкций для выбора формы страхования жизни или точна, как правила по применению полицейской дубинки.

Выбор экспертов

Вне зависимости от предметной области следующей задачей планировщиков является выбор экспертов. Главная проблема состоит в том, чтобы найти эксперта, обладающего знаниями, соответствующими выбранным целям и задачам. Так как в основе машины вывода лежит последовательное продвижение к конечной цели, то планировщик должен позаботиться о получении информации, приобретенной на основе глубокого опыта и многократных наблюдений. Для большинства экспертных систем опыт практиков полезнее теорий академиков. В системах экономического планирования, например, профессор экономики может высказать свой теоретический взгляд на определяющие факторы денежного оборота. Но такая экспертиза вряд ли заменит практический опыт эксперта, ежедневно наблюдающего за превратностями финансовой жизни корпораций. Аналогично знания фармацевта о химической динамике могут стать ценной составляющей медицинской рецептурной системы. Но эта информация не заменит опыта специалиста по внутренним болезням, накопленного путем наблюдений за реакцией пациентов на действие того или иного препарата.

Знания, необходимые для решения организационных проблем, требуют соблюдения определенной пропорции между "внутренним" опытом и "внешней" экспертизой. При построении больших систем знаний со значительным количеством предварительно определенных подзадач, часто имеет смысл сгруппировать информацию по нескольким уровням организации, от рядового персонала до администрации. Другими словами, нужно проанализировать, как развивается проблема, какую выгоду получит организация от ее решения, и кто

должен участвовать в решении данной проблемы. Допустим, что в инженерной исследовательской фирме вы возглавляете проект по созданию экспертной системы определения типа кинопленок, чтобы использовать ее при документировании различных тестовых процедур. Вы начинаете поиск знаний на самом нижнем уровне иерархии - в темной комнате техника, который "проливает свет" на характерные особенности обработки разных типов пленок. Затем вы расспрашиваете кинооператора, который обладает богатым опытом по обращению с пленкой, ее зарядкой и съемки в различных условиях тестирования. На следующем уровне, от инженеров-разработчиков вы получите информацию, касающуюся технических требований типовых тестовых процедур. Заведующий отделом сбыта снабдит вас данными относительно стоимости и спроса на пленки и наконец благодаря *Справочнику кинематографиста* вы узнаете о технических спецификациях и подходящих критериях.

Может оказаться желательным проведение внешней экспертизы для пополнения базы подходящими знаниями с информацией, полученной от других фирм. Преимущество независимого консультанта - в разнообразии опыта, накопленного различными организациями, следовательно, он обладает более глубоким пониманием предмета в области, непременно выходящей за рамки внутреннего опыта фирмы. В приведенном выше примере, внешний консультант мог бы дать дополнительные знания о новых пленках и технологиях, о которых сотрудники организации только читали в рекламных проспектах.

В том случае, когда внутренняя экспертиза ограничена или неполна, внешний консультант становится главным источником информации.

При построении небольших баз знаний с ограниченным кругом проблем, предпочтительнее один источник. Объединение знаний разных экспертов подчас затруднительно, и их рекомендации могут быть даже противоречивыми.

Выбирайте эксперта - энтузиаста; процесс извлечения знаний изнурителен.

Извлечение информации и разработка базы знаний

Помимо поиска хорошего эксперта, группа разработчиков сталкивается с проблемой эффективного извлечения информации. После выбора эксперта просто извлечение знаний менее существенно, чем извлечение правильных знаний. Для ускорения процесса перевода накопленной информации в машинную базу знаний, разработчики часто делят свои вопросы на две отдельные категории: внешние и внутренние.

Внешние вопросы устанавливают действительное содержимое базы знаний. Утверждения типа: "температура" является признаком лихорадки и значение "температуры" может быть "102 градуса по Фаренгейту" являются примерами внешнего знания. *Внутренние* вопросы, напротив, фокусируются на ходе событий - взаимосвязях между объектами, атрибутами и значениями. Утверждение "Если температура=102 градуса, то у пациента лихорадка" представляет внутреннее знание.

Передача знаний должна относиться к обеим формам его представления. В системе, основанной на правилах, мы задаем вопросы, относящиеся к внешнему знанию, для того, чтобы построить пары "объект - значение". Внутреннее знание извлекается для определения правил, которые управляют решением задачи в заданной предметной области.

Определение интерфейса пользователя

Третья составляющая передачи знаний связана с пользователем. Уже на ранней стадии разработки необходимо знать, что будет вводить конечный пользователь. Это нужно для того, чтобы убедиться, будет ли система достаточно практична и обеспечит ли максимальную совместимость со средой, в которой ей предстоит работать.

Участие пользователя выражается в следующем:

- *Конкретные задачи*. Пользователь, сталкиваясь с конкретными проблемами, может объяснить возникновение проблем и предложить возможные варианты их решения.
- *Общение*. Интерфейс пользователя должен соответствовать словарю пользователя и уровню его подготовки.

▩ *Установление связей.* Знакомство пользователя с причинами и следствиями необходимо в процессе определения взаимосвязей фактов в базе знаний.

▩ *Обратная связь.* Отличительной особенностью хорошей экспертной системы является ее способность объяснять конечному пользователю ход своих рассуждений.

Построение экспертной системы

После того как знания человека-эксперта благополучно перенесены в структурированную базу знаний, программист экспертной системы может приступать к работе. Наиболее прямой метод реализации выполнения правил - непосредственное включение их в текст программы. Более гибкий подход состоит в том, чтобы создать пополняемую систему накопления пар "объект - значение" и связи их с предложенными пользователем правилами.

В последующих главах мы исследуем специальные методы представления знаний, посмотрим как выполнять эвристические правила и проверим справедливость наших утверждений. Кроме того, мы рассмотрим компоненты машины вывода и интерфейса пользователя, а затем попробуем эти разрозненные части свести воедино. Каждая глава содержит листинги написанных на Паскале программ основных модулей экспертной системы.

РАЗДЕЛ 2

ПОСТРОЕНИЕ ЭКСПЕРТНОЙ СИСТЕМЫ

ГЛАВА 4

Представление фактов

В этой главе мы начинаем детальное изучение экспертной системы, основанной на правилах. Сначала обсудим формы представления фактов в базе знаний и определим константы, типы и переменные, использованные при разработке нашей экспертной системы. В следующих главах мы постепенно будем вводить программные модули и создадим иллюстративную программу, а по мере построения будем изучать стратегию, лежащую в основе ее разработки.

Язык Паскаль в нашем примере выбран не случайно. Во-первых, Паскаль - один из наиболее простых, а следовательно, и наиболее универсальных языков программирования высокого уровня; программы, написанные на Паскале, легко читать. Во-вторых, он обеспечивает исключительно высокую степень переносимости (даже большую, чем версии языка Бейсик для персональных компьютеров), с минимальными изменениями в словаре или синтаксисе. В-третьих, его структурное сходство с мнемоническими псевдокодами делает листинги программ на Паскале понятными практически любому программисту. В-четвертых, благодаря таким сравнительно недорогим программным продуктам как Турбо Паскаль, он доступен самым разным группам читателей. В большинстве версий Паскаля и компилятор, и выполняемый модуль включаются в один экономичный пакет.

Обзор структуры

Экспертная система объединяет три подсистемы: базу знаний, машину вывода и интерфейс пользователя. База знаний состоит из фактов и правил. Факты базы знаний описывают то, что известно о предметной области в данный момент. Правила устанавливают ситуационные, концептуальные, причинные или прецедентные взаимосвязи между этими фактами.

Следовательно, экспертная система должна иметь механизм для введения фактов и правил в базу знаний, поддерживающий набор фактических выражений и показывающий содержимое для его периодического обновления системными инженерами и операторами. В этой и следующей главах мы разработаем механизмы для решения данных задач.

Представление фактов в базе знаний

Мы представляем факты, определяя *объекты*, описывая их *атрибуты* и придавая им эквиваленты или *значения*. В экспертной системе под словом "объект" подразумеваются как физические предметы (например, "термометр" или "контракт"), так и общие представления (такие как "жар" или "цена"). С объектами связываются атрибуты, которые описывают их с нашей (и экспертной системы) точки зрения. Атрибутами объекта "термометр" могут быть: "чувствительный к температуре", "стеклянный" и "градуированный". Аналогично "температура" является атрибутом объекта "жар", а "прибыль" - атрибутом "цены". Третья единица - значение - задает точность. Например, значение температуры может быть 102° по Фаренгейту, а значение прибыли - 50% . Чтобы отличать такие фактические выражения от традиционных компьютерных данных, мы будем говорить о триплете "объект - атрибут - значение", как о *декларативных данных*.

Пары "объект-значение"

Для того чтобы упорядочить выражения фактов, можно объединить их в пары "объект - значение", соединив имя объекта с именем атрибута. Например, в триплете "термометр - температура - высокая" именем объекта будет слово "термометр", его атрибутом - "температура", а его значением - слово "высокая". Представим его в виде пары, тогда объектом будет "термометр - температура", а значением - "высокая".

Преобразование выражений фактов в пары "объект - значение" иллюстрирует табл. 4.1. Прежде чем экспертная система сможет начать обрабатывать имеющуюся в ней информацию, выражения должны быть разложены на отдельные "атомы". Рассмотрим, например, простое выражение: "Если температура высокая, то у пациента лихорадка". Данное правило подразумевает наличие следующих фактов-атомов: "У пациента температура", "Температура высокая" и "У пациента лихорадка". Пары "объект - значение" можно представить таким образом: "термометр-температура=высокая" и "пациент-симптом=лихорадка".

Таблица 4.1

Представление фактов через пары "объект - значение"

Выражение	Пары "объект - значение"
У пациента температура	пациент=имеет температуру
Температура высокая	температура=высокая
У пациента лихорадка	пациент=имеет лихорадку
Цена минимальная	цена=минимальная
Цена максимальная	цена=максимальная
Цена постоянная	цена=постоянная
Цена установлена	цена=установлена

В качестве более сложного примера рассмотрим фразу, используемую Верховным Судом США для обозначения неправильно установленной цены: "любая максимальная, минимальная, постоянная, или установленная цена". Здесь очевид-

шыми парами "объект - значение" будут: "цена=максимальная", "цена=минимальная", "цена=постоянная" и "цена=установлена".

Список объектов

Для представления объектов в базе знаний используется структура данных, называемая *сцепленным списком*. Каждая единица этого списка называется *узлом* и содержит поля, в которые заносится информация об объекте. Одно из полей служит указателем, сообщающим системе, где искать следующий сцепленный узел списка. Последний узел указывает на *nil*, это означает, что список узлов исчерпан.

Кроме того, каждый узел в списке объектов имеет второй указатель, который определяет начало списка значений, связанных с именем объекта. Этот внутренний список называется *списком значений* объектов.

На рис. 4.1 видно, каким образом каждый узел в сцепленном списке объектов указывает на следующий по порядку узел.

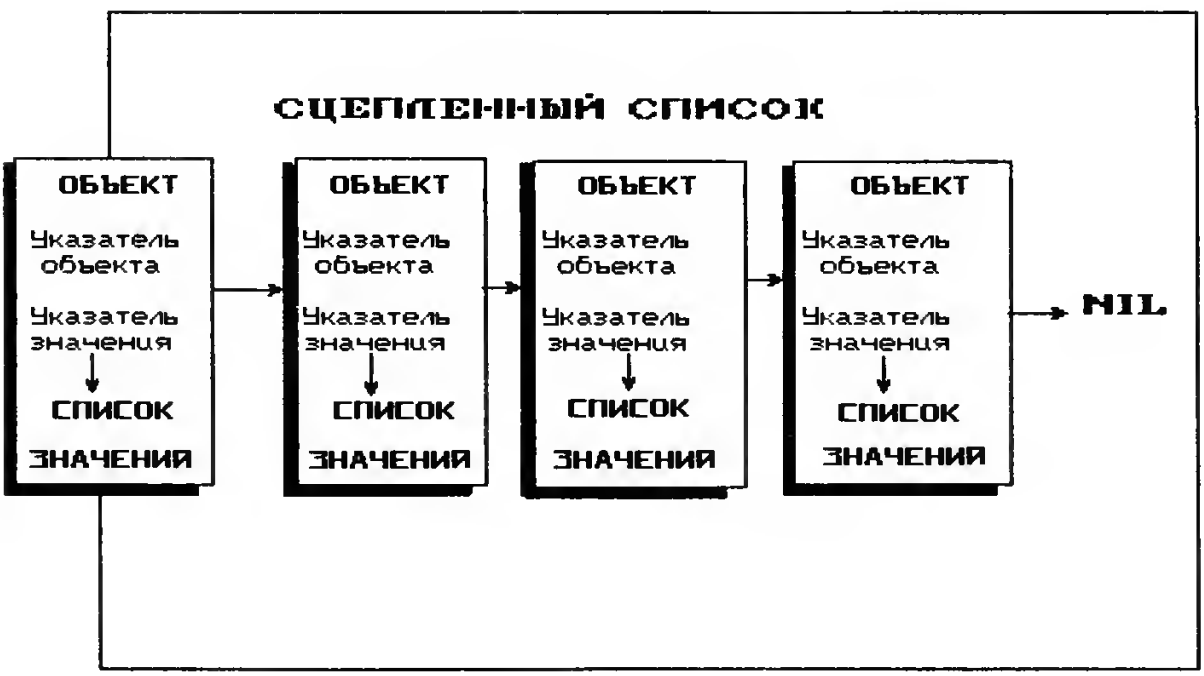


Рис. 4.1. Схема сцепленного списка объектов

Список значений

Табл. 4.2 иллюстрирует, как каждый узел поддерживает свой собственный список значений. При этом каждое предыдущее значение указывает на последующее до тех пор, пока не будет исчерпан весь список.

Здесь отдельные узлы в списке объектов содержат объекты: "возраст", "профессия" и "подчиненный". Узел "возраст" имеет только одно значение. Список значений узла "профессия" содержит три величины - "врач", "адвокат" и "индейский вождь" (причудливый анахронизм, означающий "Член Совета племени коренного населения Северной Америки").

Таблица 4.2

Пример связанного списка "объект - значение"

Список объектов	Возраст	Профессия	Подчиненный
Список значений	32	Врач Адвокат Индейский вождь	Бетти Джейн Спот

В табл. 4.3 показан аналогичный список для клинических проявлений, используемых при диагностике сердечных приступов. Узел, содержащий имя объекта "боль", имеет список, состоящий из трех значений: "грудь, "живот" и "левая рука". Пациенты, страдающие сердечными приступами, часто испытывают боль в указанных частях тела. Кроме того, врач расшифровывает электрокардиаграмму пациента (ЭКГ), пытаясь найти признаки ишемии или низкого содержания кислорода в крови. Узел списка объектов, содержащий имя "ЭКГ", имеет три связанных с ним значения: "нормально", "поднятый ST" и "отрицательный ST". Когда часть ЭКГ, называемая "зубцом ST", отрицательна, то обычно пациент уже страдает ишемией. В списке объектов, выражение факта включает все возможности - нормальный зубец, поднятый и отрицательный. Аналогично узел, содержащий имя объекта "КФК" (один из трех анализов крови, используемых для

установления диагноза), имеет три возможных значения: "нормальное", "повышенное" и "пониженное".

Таблица 4.3

Список "объект - значение": диагностика инфаркта

Объект	боль	ЭКГ	КФК
Значения	грудь живот левая рука	нормальный поднятый ST отрицательный ST	нормальное повышенное пониженное

Третий вариант списка "объект - значение" представлен в табл. 4.4. Здесь именами объектов служат критерии, используемые юристами при определении нарушений политики установления цен. Список значений каждого узла включает несколько возможных выражений.

Таблица 4.4

Список "объект - значение": определение нарушений установки цен

Объект	цена	действие	последствия
Значения	минимальная максимальная постоянная	законное незаконное	никаких штраф расторжение контракта

Определения

Теперь мы знаем как создавать список связанных объектов. На следующем шаге определим константы, типы и переменные, которые будем использовать в программе. Читателям, незнакомым с Паскалем, поясним, что *определение констант* означает ввод имен констант. Аналогично, при *определении типов* вводятся описательные имена, в отличие от числовых констант. *Объявление переменных* задает переменные программы в соответствии с ранее определенными типами.

Например, чтобы ограничить длину строки для записи имен всех объектов и значений 40 символами, мы опишем константу WORD_MAX следующим образом:

```
WORD_MAX=40;
```

Таким же образом мы можем установить максимальную длину строки в 80 символов, определив константу:

```
LINE_MAX=80;
```

В модулях программы будут использоваться следующие константы:

```
COLON=':';
PERIOD='.';
COMMA=',';
SPACE=' ';
EQUALS='=';
DEFINITE='100';
```

Далее определим типы строк, к которым относятся константы. Сначала установим максимальную длину имени объекта и значения. Затем ограничим длину вводимой строки:

```
word_string=string[WORD_MAX];
line_string=string[LINE_MAX];
```

Кроме того, нам нужно задать определения типа, необходимые для создания и манипулирования списком объектов. Для этого прежде всего создадим указатель объекта и указатель значения; они будут указывать на конкретную запись, содержащую ранее определенное значение или имя объекта. Паскаль использует для этой цели *индексный указатель*. Указатель с установленным типом ^T ссылается на анонимную переменную типа T, или на пустое значение *nil* (если значение отсутствует). В последующих определениях левая часть равенства будет представлять описание указателя, а правая часть - тип определяемой области.

```
value_ptr=^value;
object_ptr=^object;
```

Следующие определения типа создают запись, которая будет содержать имя значения до 40 символов длиной и связанный с ним указатель. Остальная часть этого описания не используется в первых модулях, которые мы создадим, но будет полезна при разработке модулей, которые мы создадим позже.

```
value=RECORD
        name:word_string;
        cert:integer;
        setby:word_string;
        next:value_ptr
END;
```

Следующее определение типа также понадобится нам позднее.

```
legal_ptr=^value;
legal_value=RECORD
        name:word_string;
        next:legal_ptr
END;
```

Теперь создадим запись, которая будет содержать имя объекта, длиной до 40 символов, указатель значения и указатель объекта. Указатель значения - это поле, указывающее на имя значения в списке значений; указатель объекта - поле, указывающее на следующий узел в текущем списке объектов. Не все эти определения будут применены немедленно, но со временем они пригодятся.

```
object=RECORD
        name:word_string;
        question:word_string;
        multivald:boolean;
        legal_list:legal_ptr;
        sought:boolean;
        value_list:value_ptr;
        next:object_ptr
END;
```

Остальные определения типа будут использоваться по мере необходимости:

```
prem_ptr=^prem;
con_ptr=^con;
rule_ptr=^rule;
prem=RECORD
        object:word_string;
        value:word_string;
        next:prem_ptr
        END;
con=RECORD
        object:word_string;
        value:word_string;
        cert:integer;
        next:con_ptr
        END;
rule=RECORD
        name:word_string;
        prem:prem_ptr;
        con:con_ptr;
        next:rule_ptr
        END;
```

Ниже приведены определения переменных, которые понадобятся нам в программе экспертной системы. Первые две будут нужны в программе меню для выбора ввода. Остальные - для управления объектами и значениями.

```
VAR
        max_choice,
        choice_lim,
        choice:integer;
        last_try,
        top_fact:object_ptr;
        s_word,
        s_object,
        s_value:word_string;
        s_line:line_string;
        s_cf:integer;
        top_rule:rule_ptr;
        rules:Text;
        explain:boolean;
```

Теперь мы определили константы, типы и переменные, необходимые для построения экспертной системы на Паскале. Полный листинг определений типов и глобальных переменных приведен в приложении А. В следующей главе мы разработаем модули программы, образующие блоки, из которых строится система.

ГЛАВА 5

Управление фактами в базе знаний

Экспертная система должна иметь механизм для ввода фактов и правил в базу знаний, поддержания набора выражений в базе знаний и вывода содержимого базы знаний для просмотра разработчиками и пользователями системы.

В этой главе мы рассмотрим требования программирования к разработке и поддержанию списка сцепленных объектов. Прежде всего, нам нужен механизм для добавления к данному списку новых узлов. Далее понадобятся средства ввода имен новых объектов и значений. И наконец, потребуется способ нахождения и извлечения конкретных имен для просмотра. Кроме того, будет полезно иметь возможность проверить, имеет ли объект определенное значение и истинен ли отдельный факт.

Для выполнения этих функций мы введем следующие модули:

MAKE_NODE обеспечивает возможность добавления к списку нового объекта.

FIND_OBJECT ищет в списке объектов имя конкретного объекта.

SPLIT извлекает из строки имя объекта и имя значения.

TEST проверяет истинность отдельного факта.

ADD_OBJECT позволяет добавлять в базу знаний имена объекта и значения.

SEE_VALS выводит все элементы списка значений любого узла.

SEE_OBJECTS выводит имена всех объектов базы знаний.

Основная программа обеспечивает интерфейс пользователя при вводе и объединяет различные программные модули при выполнении. В этой и каждой последующей главе мы будем постепенно строить основную программу для объединения новых модулей и добавления новых свойств к уже существующим модулям. В гл. 13 мы покажем более сложную перспективу законченной системы в действии.

Создание узла в списке объектов

Первым модулем программы экспертной системы является механизм для создания нового узла в списке сцепленных объектов. Следующая процедура добавляет новый узел в вершину списка объектов и устанавливает указатель на объект, поименованный в этом узле. Первый оператор после инструкции BEGIN создает новый узел. Второй определяет вершину списка, а третий устанавливает указатель вершины на вновь созданный узел. На следующем шаге указатель вновь созданного узла устанавливается на вершину списка объектов. В конце процедуры по команде WITH выполняется инициализация всех полей новой записи.

```
PROCEDURE make_node(VAR curr_object:object_ptr);
VAR
        head:object_ptr;
        BEGIN
        new(curr_object);
        head:=top_fact;
        top_fact:=curr_object;
        WITH curr_object^ DO
                BEGIN
                next:=head;
                value_list:=NIL;
                question:='';
                legal_list:=NIL;
```

```
                multivald:=FALSE;
                sought:=FALSE;
                END
        END;
```

Помещение имени объекта в сцепленный список

Следующая задача - поместить в список имя какого-либо заданного объекта. Модуль FIND_OBJECT ищет имя объекта в списке; если имя найдено, указатель устанавливается на его место в списке объектов. В противном случае, он устанавливается на *nil*.

В этой процедуре программа просматривает список объектов в поисках имени *f_object*. Найдя его, программа устанавливает указатель FIND_OBJECT на него; в противном случае, выдается значение *nil*.

```
FUNCTION find_object(f_object:word_string):object_ptr;
VAR
curr_object:object_ptr;

BEGIN
IF (last_try<>NIL) and (last_try^.name=f_object)
        THEN find_object:=last_try
ELSE
        BEGIN
        curr_object:=top_fact;
        last_try:=NIL;
        find_object:=NIL;
        WHILE ((curr_object<>NIL)AND(last_try=NIL)) DO
                BEGIN
                IF(curr_object^.name=f_object) THEN
                        BEGIN
                        find_object:=curr_object;
                        last_try:=curr_object
                        END;
                curr_object:=curr_object^.next
                END
        END
END;
```

Расщепление пары "объект - значение"

Когда выражение факта вводится в виде пары "объект - значение", система должна отличить имя объекта от имени значения. Следующий модуль, SPLIT, воздействует на выражение вида ОБЪЕКТ=ЗНАЧЕНИЕ, извлекая имена объекта и значения и выводя их в отдельные строки.

Процедура ищет во вводимой строке позицию символа "=". Затем имя объекта помещается в *f_object*, а имя значения - в *f_value*.

```
PROCEDURE split(f_line:line_string;
        VAR f_object,f_value:word_string);

VAR
st_left,
st_right:integer;

BEGIN
st_right:=pos(PERIOD,f_line);

IF (st_right=length(f_line))
        THEN f_line:=copy(f_line,1,st_right-1);
st_left:= pos(EQUALS,f_line);
st_right:=pos(COMMA,f_line);

IF ((st_left=0) AND (st_right=0)) THEN
        f_object:=f_line;
IF (st_right=0) THEN st_right:=length(f_line)+1;
IF (st_left>0) THEN
        BEGIN
        f_object:=copy(f_line,1,st_left-1);
        IF (pos(')',f_object)=0) THEN
            f_value:= copy(f_line,st_left+1,st_right-st_left-1)
        END;
st_right:=pos(')',f_object);
IF(st_right>0) THEN
        f_object:=copy(f_line,1,st_right-1)
END;
```

Проверка объектов и значений

Следующий модуль будет использоваться для проверки истинности пары "объект - значение" (наличия ее в базе знаний). Сначала программа устанавливает, содержится ли имя оговоренного объекта в списке объектов, а затем ищет имя значения. После оператора BEGIN подпрограмма вызывает модуль *find_object*.

```
FUNCTION test(f_object,f_value:word_string):value_ptr;
VAR
curr_object:object_ptr;
curr_value:value_ptr;

BEGIN
curr_object:=find_object(f_object);
test:=NIL;
IF (curr_object<>NIL) THEN
        BEGIN
        curr_value:=curr_object^.value_list;
        WHILE (curr_value<>NIL) DO
                BEGIN
                IF (curr_value^.name=f_value) THEN
                test:=curr_value;
                curr_value:=curr_value^.next
                END
        END
END;
```

Добавление объекта к списку

Следующий модуль передает две строки: имя объекта и имя значения. Эта процедура добавляет имя объекта к списку сцепленных объектов и вставляет имя значения в соответствующий список значений узла объекта.

На первом шаге проверяется наличие имени объекта в списке объектов. Если имя объекта не найдено, программа создает новый узел и добавляет указанное имя объекта в вершину списка. На следующих шагах осуществляется поиск в списке значений, и если имя значения в нем отсутствует, оно добавляется в вершину списка.

```
PROCEDURE add_object(f_object,f_value:word_string);
VAR
curr_object:object_ptr;
value_list,
head:value_ptr;

BEGIN
curr_object:=find_object(f_object);
IF (curr_object=NIL) THEN
        make_node(curr_object);
curr_object^.name:=f_object;
curr_object^.sought:=TRUE;
value_list:=test(f_object,f_value);
IF(value_list=NIL) THEN
        BEGIN
        head:=curr_object^.value_list;
        new(value_list);
        WITH value_list^ DO
                BEGIN
                next:=head;
                cert:=0;
                setby:='';
                name:=f_value
                END;
        curr_object^.value_list:=value_list
        END
END;
```

Вывод списка значений

Экспертная система должна обладать возможностью вывода полного списка всех значений и объектов базы знаний. Следующий модуль, SEE_VALS, с помощью указателя объекта выводит на экран дисплея все имена значений, содержащиеся в списке значений узла объекта. Сначала процедура находит указанный узел объекта, а затем перебирает имена в списке значений. По команде *write* программа выводит имя текущего значения на экран, а затем обращается к указателю *(p1^.next)* для определения следующего элемента. Когда устанавливается значение *nil*, вывод прекращается.

```
PROCEDURE see_vals(curr_object:object_ptr;cf_on:boolean);

VAR
curr_value:value_ptr;
cf:integer;

BEGIN
curr_value:=curr_object^.value_list;
write(curr_object^.name,EQUALS);
IF(curr_value=NIL) THEN write('Не определено');
WHILE (curr_value<>NIL) DO
        BEGIN
        write(curr_value^.name);
        IF(cf_on=TRUE) THEN
                BEGIN
                cf:=curr_value^.cert;
                write(', оп=',cf)
                END;
        curr_value:=curr_value^.next;
        IF(curr_value<>NIL) THEN write(',')
        END;
writeln
END;
```

Вывод фактов базы знаний

Следующий модуль, SEE_OBJECTS, будет использоваться для вывода на экран имен всех объектов и имен значений введенных на данный момент в базу знаний. После оператора BEGIN, процедура печатает заголовок: "ФАКТЫ БАЗЫ ЗНАНИЙ", затем сортирует имена объектов в списке объектов, выводя все имена и соответствующие им значения до тех пор, пока не будет достигнут конец списка (*nil*).

```
PROCEDURE see_objects(cf_on:boolean);

VAR
curr_object:object_ptr;

BEGIN
writeln;
writeln('ФАКТЫ БАЗЫ ЗНАНИЙ');
```

```
writeln;
curr_object:=top_fact;
WHILE(curr_object<>NIL) DO
        BEGIN
        see_vals(curr_object,cf_on);
        curr_object:=curr_object^.next
        END;
writeln;
writeln('(КОНЕЦ БАЗЫ ЗНАНИЙ)')
END;
```

Интерфейс пользователя

Последним в этой главе мы введем модуль основной программы, объединяющий предшествующие блоки с помощью интерфейса пользователя. Эта процедура позволяет оператору добавлять, проверять и выводить на экран выражения фактов базы знаний, используя пары "объект - значение". Когда оператор выбирает возможность "Добавление фактов в базу знаний", программа вызывает модуль *add_object*. Если выбирается возможность 2 - "Проверка истинности фактов", программа сравнивает вводимую пару "объект - значение" с имеющимися в базе знаний. Если же пользователь нажмет клавишу 3 - "Просмотр базы знаний", то программа вызывает модуль *see_objects* для вывода на экран полного списка имен объектов и связанных с ними списков значений.

Рис. 5.1 иллюстрирует связи между основной программой и модулями, введенными в этой главе.

Рис. 5.1. Основная программа

```
BEGIN
max_choice:=3;
choice_lim:=max_choice+1;
last_try:=NIL;
top_fact:=NIL;
choice:=0;
WHILE (choice<>choice_lim) DO
        BEGIN
        writeln;
        writeln('1. Добавление факта в базу знаний');
        writeln('2. Проверка истинности факта');
        writeln('3. Просмотр фактов базы знаний');
        writeln(choice_lim,'. Выход');
        writeln;
        write('Введите номер от 1 до ',max_choice,' и нажмите
                ENTER:');
        readln(choice);
        writeln;
        writeln;
        IF (choice>0) AND (choice<=choice_lim) THEN
                BEGIN
                CASE choice OF
                        1:
                        BEGIN
                        writeln('Какой факт вы хотите добавить в
                        базу знаний ?');
                        writeln('(Введите:
                                (ОБЪЕКТ)=(ЗНАЧЕНИЕ))');
                        readln(s_line);
                        split(s_line,s_object,s_value);
                        add_object(s_object,s_value);
                        writeln('Факт добавлен')
                        END;

                        2:
                        BEGIN
                        writeln('Какой факт вы хотите
                                проверить ?');
                        writeln('Введите:
                                (ОБЪЕКТ)=(ЗНАЧЕНИЕ)');
                        readln(s_line);
                        split(s_line,s_object,s_value);
```

```
                    IF test(s_object,s_value)=NIL THEN
                        writeln('НЕВЕРНО')
                    ELSE
                        writeln('ВЕРНО')
                    END;

                    3: see_objects(FALSE);
                    END
                END
            ELSE writeln('Вы должны ввести число от 1
                до ',choice_lim)
            END
END.
```

ГЛАВА 6

Представление знаний

Теперь экспертная система имеет возможность управлять фактами базы знаний; она может воспринимать новые факты в виде пар "объект - значение", выполнять поиск конкретного факта в базе знаний и выводить на экран список всех пар, имеющихся в базе знаний на данный момент. Для демонстрации системы в действии мы создадим небольшую базу знаний для оценки состояния здоровья и прогнозирования продолжительности жизни. В этой главе мы разберем синтаксис записи фактов и пар "объект - значение". В каждой последующей главе мы будем расширять эту систему по мере добавления к программе новых модулей.

Система определения состояния здоровья будет оценивать здоровье индивидуума на основании введенных значений заранее определенных факторов риска и влияния привычек на продолжительность жизни. Программа задаст пользователю несколько вопросов о его привычках и образе жизни. На основании ответов система предскажет продолжительность жизни индивидуума.

Так же как мы описываем каждый модуль программы и приводим его текст, мы будем добавлять новые компоненты к базе знаний. В гл. 4 мы создали систему ввода фактов в виде пар (состоящих из имен объекта и значения), а не триплетов (объект, атрибут, значение). Поэтому синтаксис требует, чтобы каждый факт вводился в таком формате:

ОБЪЕКТ=ЗНАЧЕНИЕ

Вот примеры правильного написания:

возраст=35
курение=да
вес=75

Для экономии места наша экспертная система, после завершения разработки будет при считывании фактов игнорировать пробелы между словами. Следовательно, несколько слов в левой или правой части равенства нужно объединять с помощью понятного разделителя - подчеркивания (_) или дефиса (-). Например:

курильщик_сигарет=да
холестерин-в-день=1200-грамм
ежедневные_упражнения=30_минут

Без таких разделителей имена объекта или значения, состоящие из нескольких слов, сольются в одно, напоминая невразумительное бормотание.

Наша демонстрационная система будет учитывать 12 критериев здоровья:

1. Возраст.
2. Вес, с учетом телосложения.
3. Пол.
4. Курение.
5. Потребление холестерина.
6. Потребление ненасыщенных жиров.
7. Потребление соли.
8. Потребление кальция.
9. Происхождение.
10. Расовая принадлежность.
11. Тип личности.
12. Потребление алкоголя.

Программа предложит оператору ввести значения объектов для построения в каждой области выражения фактов. Со временем, программа сможет извлекать некоторые значения из статистических характеристик, записанных в файлы базы данных. В процессе консультации компьютер будет запрашивать пользователя о его питании, привычках и типе личности.

На этот раз мы создадим серию выражений "объект -

значение", представляющих факты в базе знаний состояния здоровья. После того как мы создадим машину вывода, эти факты будут включены в эвристические правила, которые система сможет применить для "рассуждения" о состоянии здоровья индивидуума.

Давайте поочередно рассмотрим каждый критерий.

Возраст, вес и пол

Эксперт-человек, оценивающий здоровье индивидуума, сначала устанавливает его возраст, вес и пол. Значения этих объектов гораздо важнее других факторов. Например, у 60-летнего тучного мужчины уровень риска умереть от сердечного приступа намного выше, чем у 20-летней девушки с нормальным для своего роста и сложения весом.

Мы можем выразить возраст индивидуума таким образом:

возраст=60

Аналогично вес человека также можно представить парой "объект - значение":

вес=200

Очевидно, что вес зависит от таких факторов как пол, рост и телосложение. Например, вес в 100 фунтов может быть нормальным весом для женщины ростом в 5 футов 4 дюйма среднего телосложения; но тот же самый вес для 6-футового мужчины будет явно ненормальным. И наоборот, если 200 фунтов - нормальный вес для мужчины ростом 6 футов 2 дюйма, он слишком велик для женщины ростом 5 футов.

Для пола в качестве имени значения мы будем использовать сокращение:

пол=ж

Здесь важно соблюдать соответствие. Если мы используем в одном месте в качестве имени значения "м", а в другом "муж", то логика системы не сработает. Таким же образом мы можем представить рост и телосложение:

рост=5_фт
телосложение=среднее

В будущих модулях наша экспертная система будет применять правила, относящиеся к полу, весу, возрасту и телосложению, для суждения о нормальности веса индивидуума. Заключение системы будет представлено в виде выражения факта вида:

вес=недостаточный

Критерии курения и питания

Не секрет, что курение оказывает отрицательное влияние на состояние здоровья. Когда наша программа запрашивает пользователя о привычке к курению, она делает вывод относительно степени риска заболевания индивидуума сердечной недостаточностью и раком. Объект с именем КУРИЛЬЩИК может принимать значения ДА или НЕТ:

курильщик=да

Кроме того, программа оценки состояния здоровья будет рассматривать влияние четырех компонентов питания - холестерина, ненасыщенных жиров, соли и кальция. Наука утверждает, что риск сердечных заболеваний ниже у тех людей, которые употребляют пищу с низким содержанием холестерина и относительно высоким содержанием ненасыщенных жиров. Большое количество соли в пище способствует повышенному кровяному давлению, которое, в свою очередь, является фактором риска для более серьезных форм сердечных заболеваний. Потребление кальция в основном касается женщин старше 40 лет; низкое содержание кальция увеличивает риск остеопороза, особенно у пожилых женщин. С учетом этих соображений, введем следующие дополнительные факты:

потребление_холестерина=низкое
потребление_ненасыщенных_жиров=высокое

потребление_соли=низкое
потребление_кальция=высокое

Тип личности и расово-этнические критерии

Другим важным фактором здоровья является тип личности индивидуума. В известных работах психологов, обследовавших более 10000 пациентов с сердечными заболеваниями, установлено, что агрессивный, невыдержанный тип личности, "Тип А", в большей степени подвержен фатальным сердечным приступам, чем покладистый, менее амбициозный "Тип В". Поэтому наша экспертная система будет исследовать такое выражение:

тип_личности=тип-b

Происхождение и расовая принадлежность также имеют определенное значение. Чернокожий мужчина родом из Средиземноморья имеет повышенный фактор риска развития сердечных заболеваний:

район=средиземноморье
раса=негроидная

Когда машина вывода применяет правила, управляя фактами базы знаний, она может проверить, насколько обоснованы следующие факты:

риск_сердечного_заболевания=высокий
риск_остеопороза=ниже_среднего
риск_рака=выше_среднего

Потребление алкоголя также связано с продолжительностью жизни. Люди, употребляющие очень умеренное количество алкоголя (около 30 граммов в день) обычно живут дольше абсолютно непьющих и тех, кто склонен к излишествам. Поэтому добавим в базу знаний оценки здоровья следующий факт:

потребление_алкоголя=умеренное

Экспертная система решает проблему, преследуя определенную цель. В нашей демонстрационной системе эта цель выражена в форме рекомендаций. Каждое предполагаемое решение будет иметь вид выражения факта такого типа:

предсказанная_продолжительность=60_лет

Это решение показывает, что человек по предсказанию системы, основанному на анализе данных об образе его жизни и привычках, проживет до 60 лет.

Чтобы проверить систему в действии, используйте редактор для ввода программных модулей, описанных в гл. 4 и 5. Затем скомпилируйте программу и запустите ее. При запуске программы на экране появится меню:

1. Добавление факта в базу знаний
2. Проверка истинности факта
3. Просмотр фактов базы знаний
4. Выход

Введите номер от 1 до 4 и нажмите ENTER:

Для ввода выражений фактов, перечисленных в этой главе, вам нужно выбрать возможность 1. Для проверки факта - возможность 2. После того как вы ввели последний факт, можно выбрать возможность 3 и просмотреть все содержимое базы знаний.

Пробный запуск программы

Давайте сделаем пробный запуск нашей программы, чтобы посмотреть, что же мы имеем. Выполните программу, как описано выше. Когда появится меню, выберите возможность 1 для добавления пар "объект - значение". Программа выведет на экран дисплея:

Введите номер от 1 до 4 и нажмите ENTER:1

Какой факт вы хотите добавить в базу знаний?
(Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ))

Программа предлагает вам ввести факт в виде пары "объект - значение".

возраст=50
Факт добавлен.

1. Добавление факта в базу знаний
2. Проверка истинности факта
3. Просмотр фактов базы знаний
4. Выход

Введите номер от 1 до 4 и нажмите ENTER:

Программа добавляет выражение факта в базу знаний и снова выводит на экран меню. Теперь выберите возможность 2 и посмотрите, истинен ли факт, который вы только что добавили.

Введите номер от 1 до 4 и нажмите ENTER:**2**

Какой факт вы хотите проверить?
(Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ))

Введите ту же самую пару "объект - значение", которую вы только что добавили в базу знаний:

возраст=50
ИСТИННО

1. Добавление факта в базу знаний
2. Проверка истинности факта
3. Просмотр фактов базы знаний
4. Выход

Введите номер от 1 до 4 и нажмите ENTER:

Система проверяет факт и выдает ИСТИННО. Выберите возможность 3 для просмотра содержимого всей базы знаний.

Введите номер от 1 до 4 и нажмите ENTER:**3**

ФАКТЫ БАЗЫ ЗНАНИЙ:
возраст=50

ГЛАВА 7

Представление неопределенности

В двух предыдущих главах было показано как экспертная система управляет вводом выражений фактов в виде пар "объект - значение". В этой главе мы рассмотрим как система использует выражения неопределенной информации.

Коэффициенты определенности

Поскольку не все знания строго определены, экспертная система должна иметь средства обработки различной степени доверия в заданных выражениях фактов. Субъективные утверждения склонны к абсолютности; в заявлении "Мой любимый писгель - Достоевский" не содержится сомнений в определенности факта. Однако в выражении "Достоевский - величайший в мире романист" степень доверия более перемешна.

Таблица 7.1 иллюстрирует различный характер выражений фактов. Мы можем с абсолютной определенностью утверждать что "Фамилия пациента - Фингербендер". Если клинические показания убедительно свидетельствуют, что "У пациента коронарный эмболизм", наш уровень уверенности высок. Если же, наоборот, свидетельства того, что "Миокард ишемичен" не очень убедительны, то уровень уверенности низок.

Таблица 7.1.

Примеры коэффициентов определенности

Факт	Доверие	Определенность
Имя пациента-Фингербендер.	Определенно	100
У пациента коронарный эмболизм.	Убедительно подтверждено клиническими показаниями	90
Миокард ишемичен	Слабо подтверждено клиническими показаниями	25

Экспертные системы выражают относительную уверенность в факте с помощью коэффициентов доверия *. Коэффициент доверия (кд), равный 100, означает высший (абсолютный) уровень уверенности, коэффициент 0 - самую низкую определенность. Способы оценки определенности варьируются от стандартной теории вероятностей до совершенно субъективных оценок.

Оценка определенности

В табл. 7.2 показано использование статистики в представлении неопределенности. В этом примере цель экспертной системы - определить оптимальное распределение потока товаров по ряду торговых предприятий в зависимости от потребностей конкретных районов. За пятилетний период в районе 1 насчитывалось 30% продаж в течение 80% времени; в районе 2 - 20% продаж в течение 60% времени и т.д. Методы улучшения точности оценок включают экспоненциальное

* В литературе для обозначения недоопределенности информации применяются различные выражения: коэффициент доверия, коэффициент определенности и т.п. В дальнейшем мы будем использовать в основном определение "коэффициент доверия", так как его сокращенное обозначение - "кд" - создает определенные удобства при записи правил (его трудно спутать с началом какого- либо слова). - *Примеч. пер.*

сглаживание, показатели тенденции для учета весов сезонных или самых последних данных, пуассоновский анализ и линейный регрессионный анализ. Например, во многих системах коэффициент определенности 80 означает 80% - ную вероятность, что выражение истинно.

Таблица 7.2

Пример использования коэффициентов определенности для выражения вероятности

Территория	Процент продаж	Определенность
Район 1	30	80
Район 2	20	60
Район 3	15	75
Район 4	15	80

Но зачастую выражения фактов не так просто описать количественно или статистически, и для оценки определенности требуются более субъективные методы. Рассмотрим следующий пример: продавец компьютеров предполагает приобрести новую линию программного обеспечения. Хотя он может воспользоваться многими статистическими инструментами, большинство соображений, влияющих на решение о закупке, носят субъективный характер. Некоторые из таких критериев, с описанием четырех основных уровней определенности, которые каждый из них может принимать, перечислены в табл. 7.3.

Продавец рассматривает каждый критерий относительно четырех описательных категорий, затем приписывает коэффициент определенности соответствующей паре "объект - значение". Допустим, например, что цена потенциальной закупки ПРОДУКТ1 примерно такая же, что и у других продуктов аналогичного качества. Оценка ПРОДУКТА1 по отношению "цена/качество" - "посредственная" и продавец присваивает паре "ПРОДУКТ1=КОНКУРЕНТНАЯ_ЦЕНА" коэффициент доверия, равный 45. Допустим, ПРОДУКТ1 будет существенно способствовать продаже имеющихся у продавца продуктов. Для критерия "влияние на продажу имеющихся линий" оценка будет "отличная", и паре "ПРОДУКТ1=ДОПОЛ-

НЯЮЩИЙ" продавец присвоит коэффициент доверия, равный 90.

Эта матрица служит двум целям: во-первых, она определяет формат для оценки определенности на основе субъективных суждений; во-вторых, облегчает перевод важных фактов в пары "объект - значение", которые могут обрабатываться экспертной системой.

Программные модули для обработки переменной определенности

Модули, рассматриваемые в данной главе, позволят программе нашей экспертной системы обрабатывать переменную определенность. Применяя матричную систему оценки, аналогичную табл. 7.3, программа сможет считывать целое значение коэффициента доверия (кд) в пару "объект - значение", или наоборот, переводить субъективные выражения "плохое, среднее, хорошее, абсолютное" в числовой коэффициент определенности по шкале от 0 до 100. Дополнительные модули будут использоваться для добавления целого кд к сцепленному списку объектов и позволят изменять существующий кд.

Введем следующие программные модули:

GET_CF будет использоваться для извлечения коэффициента доверия из пары "объект - значение".

BLEND позволит изменять существующий коэффициент доверия, объединяя новое значение с ранее введенной величиной кд.

ADD_CF будет записывать новый коэффициент доверия в список "объект - значение".

Кроме того, мы изменим основную программу для подключения вновь введенных модулей.

Извлечение коэффициента доверия

Первый модуль отыскивает коэффициент доверия в строке "объект - значение" в виде:

|ОБЪЕКТ|=|ЗНАЧЕНИЕ|,кд=|КОЭФФИЦИЕНТ ДОВЕРИЯ|

Теперь, когда пользователи добавляют новые факты к базе знаний, они будут писать запятую, "кд=", и коэффициент

Таблица 7.3.

Пример использования коэффициентов определенности в выражениях субъективной оценки

Критерий	Определенность			
	Отличная (76–100)	Хорошая (51–75)	Посредственная (26–50)	Плохая (0–25)
Отношение "цена/качество"	Оценка ниже конкурирующих продуктов аналогичного качества	Оценка ниже большинства конкурирующих продуктов аналогичного качества	Оценка примерно такая же, как и у большинства продуктов анапогичного качества	Оценка выше,чем у большинства продуктов аналогичного качества
Впияние на сбыт существующих серий	Определенно будет способствовать сбыту существующих серий	Может ломочь сбыту существующих серий	Не окажет влияние на сбыт существующих серий	Может составить конкуренцию ипи помешать сбыту существующих серий
Документация	Удобные в освоении руководства, написанные ясным языком	Исчерлывающие руководства, содержащие наиболее важные сведения	Слабо написанные руководства, рассчитанные на доработку	Отрывочная, неполная или небрежно подготовпенная документация
Дополнительный сервис	Независимое от среды, снабженное лодсказками и с хорошей реакцией	Зависящее от среды, с удовлетворительной реакцией	Зависящее от среды, со случайной реакцией	Не рассчитанное на специальное приложение
Рынок сбыта	Широкий круг пользователей, быстро растущий рынок	Широкий круг лользователей, стабипьный рынок	Ограниченный рынок с потенциальными возможностями	Необеспеченный рынок с неустановленным лотенциалом
Совместимость с техническими средствами	Может устанавпиваться на вылускаемом оборудовании	Может устанавливаться на вылускаемом оборудовании, но лотребует выпуска дололнительных изделий	Может устанавливаться на некоторых видах вылускаемого оборудования	Потребует вылуска новых технических средств
Знания и лерсонап	Поддерживать лродукт может существующий персонал	За малым исключением, лоддерживать лродукт может существующий лерсонал	Потребуется некоторое дололнительное знание и обновление персонала	Потребуется существенно новое знание и новый персонал

доверия от 0 до 100. В нашей программе 0 будет представлять низший уровень определенности, а 100 - наивысший. В табл. 7.4 приведено несколько примеров правильного ввода пар "объект - значение" с соответствующими коэффициентами доверия.

Таблица 7.4

Выражения коэффициентов доверия в парах объект-значение

Объект	Значение	Доверие	Формат
Продажи	Максимум	80	продажи=максимум,кд=100
Миокард	Ишемичен	90	миокард=ишемичен,кд=90
Вкус	Кислый	75	вкус=кислый,кд=75
Область	Медицина	40	область=медицина,кд=40

Мы также оборудуем экспертную систему механизмом перевода субъективных оценок из лингвистических выражений в целые значения.

Эта функция обеспечивает поиск во введенной строке коэффициента доверия, кд. Если этот коэффициент представлен числом, функция передает его. Если же он выражен словом "плохой", функция передает значение 25. Аналогично слово "средний" преобразуется в 50, 75 обозначает "хороший", а 100 вводится для слова "отличный".

```
FUNCTION get_cf(f_line:line_string):integer;

VAR
result,
st_right,
cf:integer;
trim:line_string;

BEGIN
cf:=DEFINITE;
st_right:=pos(PERIOD,f_line);
IF st_right=length(f_line) THEN f_line:=copy(f_line,1,st_right-1);
st_right:=pos('кд',f_line);
IF (st_right>0) AND (st_right+3<LINE_MAX) THEN
        BEGIN
        trim:=copy(f_line,st_right+3,length(f_line)-st_right-2);
```

```
        val(trim,cf,result);
        IF (result>0) THEN cf:=DEFINITE;
        IF pos('плохой',trim)>0                    THEN cf:=25;
        IF pos('средний',trim)>0                   THEN cf:=50;
        IF pos('хороший',trim)>0                   THEN cf:=75;
        IF pos('абсолютный',trim)>0          THEN cf:=DEFINITE
        END;
get_cf:=cf
END;
```

Изменение коэффициента доверия

Следующий модуль позволяет изменять существующие коэффициенты доверия, объединяя текущее значение кд с новым значением, введенным оператором для того же самого факта. Чтобы объединить два имеющихся числа, применяется простой метод сглаживания. Каждое целое умножается на 100, а произведения складываются. Кроме того, сами числа перемножаются и результат вычитается из первой суммы. Для получения нового коэффициента доверия разность делится на 100.

Допустим, что оператор уже присвоил факту базы знаний коэффициент доверия, равный 50. Позже этому же факту добавляется коэффициент доверия, равный 75. Программа обрабатывает эти два числа следующим образом:

```
[(50*100)+(75*100)-(50*75)]/100=87.5
```

```
FUNCTION blend(cf1,cf2:integer):integer;

BEGIN
        blend:=((100*cf1)+(100*cf2)-(cf1*cf2)) DIV 100
END;
```

Сохранение коэффициента доверия

Следующий модуль - это процедура занесения введенного коэффициента доверия в соответствующий узел списка объектов. Сначала процедура расщепляет строку на три переменные, представляющие имя объекта, имя значения и коэффициент доверия. Затем она проверяет наличие имени объекта в списке объектов. После этого программа обращается к

списку значений, чтобы проверить, существует ли уже имя данного значения. И наконец, она добавляет к этому списку новый коэффициент доверия. Если коэффициент доверия для указанного значения в узле объекта уже задан, программа объединяет два значения.

```
PROCEDURE add_cf(f_object,f_value:word_string;cf2:integer);

VAR
cf1:integer;
curr_value:value_ptr;

BEGIN
curr_value:=test(f_object,f_value);
cf1:=curr_value^.cert;
curr_value^.cert:=blend(cf1,cf2)
END;
```

Изменение основной программы

Далее мы изменим основную программу для включения вновь созданных модулей. В первую очередь нужно изменить вывод на экран для запроса у оператора выражения факта и ввести дополнительный шаг для запуска процедуры ADD_CF, сохраняющей введенный коэффициент доверия.

```
BEGIN
max_choice:=3;
choice_lim:=max_choice+1;
last_try:=NIL;
top_fact:=NIL;
choice:=0;
WHILE (choice<>choice_lim) DO
BEGIN
    writeln;
    writeln('1. Добавление факта в базу знаний');
    writeln('2. Проверка истинности факта');
    writeln('3. Просмотр фактов базы знаний');
    writeln(choice_lim,'. Выход');
    writeln;
    write('Введите номер от 1 до ',max_choice,' и нажмите
    ENTER:');
```

```
    readln(choice);
    writeln;
    writeln;
    IF (choice>0) AND (choice<=choice_lim) THEN
        BEGIN
        CASE choice OF
            1:
            BEGIN
            writeln('Какой факт вы хотите добавить в
            базу знаний ?');
            writeln('(Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ)
            ,кд=(ЦЕЛОЕ))');
            readln(s_line);
            s_cf:=get_cf(s_line);
            split(s_line,s_object,s_value);
            add_object(s_object,s_value);
            add_cf(s_object,s_value,s_cf);
            writeln('Факт добавлен')
            END;

            2:
            BEGIN
            writeln('Какой факт вы хотите проверить?');
            writeln('Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ)');
            readln(s_line);
            split(s_line,s_object,s_value);
            IF test(s_object,s_value)=NIL THEN
                writeln('НЕВЕРНО')
            ELSE
                writeln('ВЕРНО')
            END;

            3: see_objects(TRUE);
            END
        END
    ELSE writeln('Вы должны ввести число от 1 до',choice_lim)
END
END.
```

Пробный запуск программы

Скомпилировав новые модули и переработанную основную программу, вы сможете вводить пары "объект - значение" с добавленным к ним коэффициентом доверия. Этот кд можно представить целым числом от 0 до 100, где 0 означает самый низкий уровень доверия, а 100 - наивысший. Кроме чисел, для обозначения относительного доверия к определенности выражения факта, оператор может вводить любое из слов "плохой", "средний", "хороший" или "абсолютный".

Чтобы продемонстрировать систему в действии, давайте продолжим разработку программы оценки состояния здоровья, которую мы начали в гл. 6. Скомпилируйте новые модули и изменения, а затем запустите программу *expert*. Введите каждое из выражений фактов, которые вы создали в прошлой главе, на сей раз добавляя коэффициент доверия. Ниже приводятся примеры:

возраст=35, кд=100
вес=70, кд=100
рост=150, кд=100
телосложение=среднее,кд=100
комплекция=тучная, кд=75
курение=да, кд=100
потребление_холестерина=высокое, кд=75
потребление_ненасыщенных_жиров=высокое, кд=75
потребление_соли=низкое, кд=50
тип_личности=тип_b
коронарный_риск=средний, кд=75
потребление_алкоголя=умеренное, кд=100

Запустите программу и на экране появится то же самое меню, что и в гл. 6. Однако, теперь выбрав возможность 1 для ввода факта в базу знаний, вы сможете добавить коэффициент доверия, используя формат:

ОБЪЕКТ=ЗНАЧЕНИЕ, КД=ЦЕЛОЕ

Вместо целого значения вы можете подставить описание "абсолютный", "средний", "хороший" или "плохой". Если вы опустите в выражении коэффициент доверия, то по умолчанию ему будет присвоено значение 100.

Теперь, когда вы выберете возможность 3 для вывода фактов базы знаний на экран, все выражения в списке будут включать в себя коэффициенты доверия.

ГЛАВА 8

Использование многозначных выражений

В данный момент наша экспертная система содержит две основные группы модулей одну - для управления фактами, другую - для управления коэффициентами доверия. Кроме того, основная программа обеспечивает краткий интерфейс оператора.

Сейчас программа может воспринимать строки, в которых записаны пары "объект - значение" с соответствующими коэффициентами доверия, добавлять новые узлы к сцепленному списку, отыскивать существующие имена объектов и имена значений, а также выводить на экран списки как объектов, так и значений.

В настоящей главе мы доработаем систему, чтобы получить возможность управлять списком значений каждого узла объекта, проверять, может ли конкретный объект иметь более одного значения перед тем как добавить к списку значений объекта имена множественных значений.

Многозначный объект

Точно так же как некоторые пары "объект - значение" могут быть выражены со 100%-ной определенностью, некоторые объекты в базе знаний имеют лишь одно-единственное значение. Рассмотрим, например, такое утверждение: "Мое любимое вино - шампанское". Хотя у человека есть целый ряд привязанностей, и он мог бы перечислить их в убывающем порядке, любимым может быть только один объект. И наоборот, объект "домашние_любимцы" может иметь множество значений со 100%-ной определенностью. Следовательно, узел объекта с именем МОИ_ЛЮБИМЦЫ - многозначный, тогда как узел ЛЮБИМОЕ_ВИНО - нет.

В табл. 8.1 приведен ряд других примеров. Здесь представлены имена нескольких объектов, которые могут быть использованы в базе знаний для определения конфигураций интегрированных компьютерных систем специального назначения. Объект СРЕДА может иметь только одно значение. Например, средой может быть либо учреждение, либо фабрика. И наоборот, объект ПРИМЕНЕНИЕ может иметь несколько различных значений, например, подготовка текстов, ведение баз данных, расчеты и управление процессом. Аналогично много значений у объекта ПОЛЬЗОВАТЕЛЬ (управляющий, служащий, машинистка и т.п.), так как компьютером могут пользоваться несколько человек. Но объект РЕСУРСЫ может иметь только одно значение, поскольку ограничение на ресурсы у каждого компьютера свое.

Другой пример приведен в табл. 8.2. Здесь перечислены некоторые имена объектов, которые могут быть применены в экспертной системе для диагностики различных форм сердечных заболеваний. Пациент может иметь только одну фамилию и возраст. Но характеристики ЭКГ пациента могут быть различными: "норма", "поднятый ST", или "отрицательный ST".

Таблица 8.1

Примеры однозначных и многозначных выражений

Однозначные	Многозначные
Среда	Применение
Ресурсы	Пользователь
Максимальная стоимость	Тип принтера
Максимальный объем документа	Бюджет

Аналогично у ферментного анализа крови много разных значений (в данном случае, "КФК", "ЛДГ" и "ТМ" относятся к ферментному анализу крови, используемому для подтверждения диагноза инфаркт миокарда). Но пациент может иметь только один пол и один вес. Таким образом, тесты ЭКГ, КФК, ЛДГ и ТМ многозначны, а антропометрические данные - нет.

Таблица 8.2

Некоторые примеры однозначных и многозначных объектов и списки их значений

Однозначные	Фамилия Возраст Пол Вес	Финкельбендер 65 Мужской 75
Многозначные	ЭКГ КФК ЛДГ ТМ	норма, поднятый ST, отрицательный ST норма, повышенное, пониженное норма, повышенное, пониженное норма, повышенное, пониженное

Экспертная система должна уметь обнаруживать противоречия и избыточность в базе знаний, отличая многозначные объекты от тех, которые таковыми не являются. Обеспечить ей эту возможность должен оператор - человек.

Программные модули для многозначных выражений

Модули, описываемые в этой главе, применяются для выяснения, является ли объект в сцепленном списке многозначным, и для определения его как многозначного объекта с конкретным именем.

OK_ADD определяет статус любого заданного имени объекта в списке для выяснения, многозначный он или нет.

MAKE_MULTI меняет статус объекта с "немногозначный" на "многозначный".

Проверка статуса многозначности

Сначала мы создадим программный модуль, который перед добавлением значения к списку будет контролировать, может ли быть данному объекту присвоено значение. Эта функция просматривает список объектов для проверки наличия имени данного объекта. Если объект не многозначный и

если у него есть значение с определенностью 100, программа выдает значение FALSE (т.е. данному объекту может быть присвоено только одно значение). Если объект многозначный, или определенность значения меньше 100, программа выдает значение TRUE.

```
FUNCTION ok_add(f_object:word_string;cf:integer):boolean;
VAR
curr_object:object_ptr;
curr_value:value_ptr;
is_100:boolean;

BEGIN
ok_add:=TRUE;
is_100:=FALSE;
curr_object:=find_object(f_object);
IF (curr_object<>NIL) THEN
        BEGIN
        curr_value:=curr_object^.value_list;
        WHILE (curr_value<>NIL) DO
                BEGIN
                IF (curr_value^.cert=DEFINITE) THEN
                        is_100:=TRUE;
                curr_value:=curr_value^.next
                END
        END;
IF ((cf=DEFINITE) AND (is_100=TRUE)
        AND(curr_object^.multivald=FALSE))
        THEN ok_add:=FALSE
END;
```

Объявление объекта многозначным

Следующий модуль позволяет оператору определить заданный объект как многозначный, давая возможность приписать указанному имени объекта более одного значения. Процедура MAKE_MULTI воздействует на имя объекта и меняет его статус на многозначный.

На первом шаге просматривается список объектов для проверки наличия в нем указанного имени. Если программа не находит имя объекта (curr_object), она создает новый узел и добавляет имя в вершину списка, вызывая модуль

MAKE_NODE. И наконец, эта процедура меняет значение поля MULTIVALD на TRUE.

```
PROCEDURE make_multi(f_line:line_string);

VAR
curr_object:object_ptr;
dummy,
f_object:word_string;

BEGIN
split(f_line,f_object,dummy);
curr_object:=find_object(f_object);
IF (curr_object=NIL) THEN make_node(curr_object);
curr_object^.name:=f_object;
curr_object^.multivald:=TRUE
END;
```

Изменение основной программы

Теперь, когда наша экспертная система имеет возможность проверять, является ли объект многозначным и давать статус многозначности определенным именам объектов, нужно изменить основную программу, чтобы включить в нее эти новые возможности.

В приведенном тексте программы вставлены два шага. В подпрограмме для возможности 1 перед добавлением пары "объект - значение", функция OK_ADD проверяется на значение TRUE:

```
IF (ok_add(string1)=TRUE) THEN
    BEGIN
```

Кроме того, для объявления объекта многозначным добавлена новая возможность выбора - 4. Это изменение отражено в процедуре меню командой *writeln*:

```
writeln('4. Объявление объекта многозначным');
```

и разделом CASE.....OF добавлением подпрограммы:

```
writeln('какой факт');
```

```
        writeln('вы хотите установить?');
        readln(string1);
        make_multi(string1);
```

Последний шаг в подпрограмме состоит в вызове ранее созданной процедуры MAKE_MULTI.

Ниже приводится текст основной программы, измененной для того, чтобы включить эти новые модули.

```
BEGIN
max_choice:=4;
choice_lim:=max_choice+1;
last_try:=NIL;
top_fact:=NIL;
choice:=0;
WHILE (choice<>choice_lim) DO
     BEGIN
     writeln;
     writeln('1. Добавление факта в базу знаний');
     writeln('2. Проверка истинности факта');
     writeln('3. Просмотр фактов базы знаний');
     writeln('4. Объявление объекта многозначным');
     writeln(choice_lim,'. Выход');
     writeln;
     write('Введите номер от 1 до ',max_choice,' и нажмите
     ENTER:');
     readln(choice);
     writeln;
     writeln;
     IF (choice>0) AND (choice<=choice_lim) THEN
          BEGIN
          CASE choice OF
               1:
               BEGIN
               writeln('Какой факт вы хотите');
               writeln('добавить в базу знаний ?');
               writeln('(Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ)
               кд=(ЦЕЛОЕ))');
               readln(s_line);
               s_cf:=get_cf(s_line);
               split(s_line,s_object,s_value);
               IF (ok_add(s_object,s_cf)=TRUE) THEN
```

```
                BEGIN
                add_object(s_object,s_value);
                add_cf(s_object,s_value,s_cf);
                writeln('Факт добавлен')
                END
            ELSE
                BEGIN
                write('Добавление не разрешено.');
                write(s_object);
                writeln('не многозначный');
                END
            END;
            2:
            BEGIN
            writeln('Какой факт вы хотите проверить ?');
            writeln('Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ)');
            readln(s_line);
            split(s_line,s_object,s_value);
            IF test(s_object,s_value)=NIL THEN
                writeln('НЕВЕРНО')
            ELSE
                  writeln('ВЕРНО')
            END;
            3: see_objects(TRUE);

            4:
            BEGIN
            writeln('Какой объект');
            writeln('вы хотите объявить  многозначным ?');
            writeln('(Введите:(ОБЪЕКТ))');
            readln(s_object);
            make_multi(s_object)
            END

            END
        END
    ELSE writeln('Вы должны ввести число от 1 до ',choice_lim)
    END
END.
```

Пробный запуск программы

Скомпилировав новые модули и запустив программу, вы заметите, что теперь меню содержит четвертую возможность - "Объявление объекта многозначным". Пока с помощью этой подпрограммы имя объекта не будет определено как многозначное, система будет разрешать добавлять к этому объекту только одно значение.

Чтобы проверить программу с этим новым дополнением, давайте расширим нашу гипотетическую систему оценки состояния здоровья. В предыдущих главах мы вводили факты в базу знаний в виде пар "объект - значение" с соответствующими коэффициентами доверия. Теперь система позволяет нам вводить для одного имени объекта несколько значений.

Введите факты, использовавшиеся в гл. 6. После каждого ввода выбирайте возможность 4 для объявления объекта многозначным. Затем добавьте факты, которые придают имени объекта альтернативные значения. Для просмотра содержимого базы знаний выберите возможность 3.

Прежде чем записывать каждое возможное значение для возраста, веса и роста, мы можем сгруппировать пары "объект - значение" по общим категориям или диапазонам. У каждой категории определенность равна 100, так что коэффициент доверия можно опустить (по умолчанию он равен 100).

```
возраст=27_или_меньше
возраст=между_27_и_50
возраст=50_или_больше
```

Эти диапазоны могут быть менее обширными, мы выбрали такие широкие, но статистически значительные группы для простоты демонстрации. Аналогично мы можем сгруппировать по категориям вес:

```
вес=40_или_меньше
вес=между_40_и_60
вес=между_60_и_80
вес=между_80_и_100
вес=100_или_более
```

Вес будет оцениваться в связи с другими факторами, каждый из которых также имеет диапазон возможных значений:

рост=5_футов_или_меньше
рост=между_5_футами_и_5_с_половиной_футов
рост=между_5_с_половиной_футов_и_6_футами
рост=6_футов_или_больше

комплекция=мелкая
комплекция=средняя
комплекция=крупная

После завершения разработки, экспертная система будет выводить заключение на основе перечисленных фактов о весе человека. Вывод системы будет представлен выражениями:

вес=недостаточный
вес=нормальный
вес=излишний

Когда система запрашивает оператора о привычке к курению, возможны два факта:

курение=да
курение=нет

Наша демонстрационная система анализирует также потребление холестерина, ненасыщенных жиров, соли и кальция. Ранжируя возможные значения каждого объекта, мы можем построить следующие выражения:

потребление_холестерина=высокое
потребление_холестерина=нормальное
потребление_холестерина=низкое

потребление_ненасыщенных_жиров=высокое
потребление_ненасыщенных_жиров=нормальное
потребление_ненасыщенных_жиров=низкое

потребление_соли=высокое
потребление_соли=нормальное
потребление_соли=низкое

потребление_кальция=высокое
потребление_кальция=нормальное
потребление_кальция=низкое

Система контроля здоровья будет также учитывать тип личности субъекта. В соответствии с исследованиями Фридмана-Росса, которые мы примем за основу, возможны два значения типа. Поскольку оценка типа является в значительной степени условной, доверие к каждому факту определяется коэффициентом 75.

тип_личности=тип_a, кд=75
тип_личности=тип_b, кд=75

При построении заключения системы будут учитываться такие дополнительные факты:

риск_сердечных_заболеваний=выше_среднего
риск_сердечных_заболеваний=средний
риск_сердечных_заболеваний=ниже_среднего

риск_рака=выше_среднего
риск_рака=средний
риск_рака=ниже_среднего

риск_остеопороза=отсутствует
риск_остеопороза=умеренный
риск_остеопороза=высокий

Потребление человеком алкоголя также попадает в одну из трех статистически определенных категорий:

потребление_алкоголя=нет
потребление_алкоголя=умеренно
потребление_алкоголя=чрезмерно

Поскольку система должна быть способна находить более одного решения, объект ПРЕДСКАЗАННАЯ_ПРОДОЛЖИТЕЛЬНОСТЬ имеет такие возможные значения:

предсказанная_продолжительность=84
предсказанная_продолжительность=82
предсказанная_продолжительность=80
предсказанная_продолжительность=72
предсказанная_продолжительность=70
предсказанная_продолжительность=68
предсказанная_продолжительность=64
предсказанная_продолжительность=60
предсказанная_продолжительность=56
предсказанная_продолжительность=52
предсказанная_продолжительность=50

Чтобы проверить ваши новые модули, откомпилируйте и запустите программу в исправленном варианте. Исходное меню будет содержать новую возможность - выбор 4, "объявление объекта многозначным". Выберите один из набора фактов этого раздела и добавьте к базе знаний первую пару "объект - значение". Затем выберите возможность 4 и введите имя объекта. Теперь вы можете добавить к набору оставшиеся пары*.

---

* Здесь авторы смешивают два разных понятия: многозначного объекта и разрешенных значений объекта. В начале главы упоминалось, что многозначным называется объект, который может иметь одновременно несколько абсолютно достоверных значений (например, мои_домашние_животные=кошка, собака). В противоположность этому существуют разрешенные значения, которые перечисляются в списке, и объект может принимать лишь одно из них. Так, объекту ВОЗРАСТ может быть присвоено лишь одно достоверное значение из приведенных в данном разделе. – *Примеч. пер.*

ГЛАВА 9

Вопросы и разрешенные значения

Снабдив экспертную систему механизмом для управления фактами базы знаний, мы сталкиваемся с проблемой получения осмысленной информации от пользователя. В этой главе мы остановимся на методах определения возможностей ввода и построения системы опроса. До сих пор программа распознавала пары "объект - значение" с приписанным коэффициентом доверия. Пока еще не шла речь о каких-либо "разрешенных значениях" для данного выражения факта.

Модули, описанные в данной главе, будут использоваться для извлечения и сохранения имен разрешенных для определенного объекта значений и построения вопросов, которые будет задавать система при поиске значения объекта.

Разрешенные значения

Для пояснения концепции разрешенных значений, обратимся к гипотетической консультации, полученной пленником космического корабля пришельцев от экспертной системы освобождения. Один из вопросов, предлагаемый системой пользователю, касался освещения, проникающего сквозь иллюминатор. Ответ на вопрос был ограничен двумя возможностями: прямой и непрямой. Это и есть *разрешенные значения* объекта "солнечный_свет".

В качестве другого примера представим себе экспертную систему, предназначенную для того, чтобы посоветовать энергичному и перспективному молодому специалисту, как ему себя держать с молодой особой, с которой он познакомился. Когда наш молодой человек консультируется с экспертной системой, программа задает ему вопросы такого типа:

Какое образование получила девушка?
1. Массачусетский Технологический институт
2. Кулинарный колледж
3. Школа водителей грузовиков
3

Во что одета молодая леди?
1. Мантия до пола
2. Бикини из шнурочка
3. Костюм миди
3

Чем занимается девушка?
1. Библиотекарь
2. Антивоенный активист
3. Нейтронный спектроскопист
2

В этой консультации выбор одного из возможных ответов представляет одно из возможных значений объекта, к которому относится вопрос. Первый вопрос касается объекта *образование*, и его разрешенные значения: *Массачусетский Технологический институт, кулинарный колледж и школа водителей грузовиков*. Следующий вопрос перечисляет разрешенные значения объекта *одежда*, а третий - объекта *занятие*.

Следовательно, нашей экспертной системе требуется механизм считывания имен разрешенных значений из вводимой строки и занесения их в сцепленный список.

Вопросы о разрешенных значениях

Как видно из предыдущей консультации, программа должна также иметь возможность задавать вопросы, относящиеся к определенному объекту. Когда пользователь вводит разрешенные значения объекта, каждое поименованное значение становится одной из возможностей, выбираемых в ответ на вопрос. Оператор должен ввести текст этого вопроса.

В табл. 9.1 приведено несколько примеров имен объектов, разрешенных значений и вопросов экспертной системы.

Таблица 9.1.

Примеры разрешенных значений

Объект	Разрешенные значения	Вопрос
ЭКГ	норма, поднятый ST, отрицательный ST	Как выглядит зубец ST на ЭКГ пациента?
цена	неограничена, минимальна, максимальна, статична	Как определена цена в торговом договоре?
класс	автомобиль, передвижной дом, недвижимость	Какой вид собственности вы предпочитаете в качестве обеспечения?
цвет	красный, белый	Какого цвета вина вы предпочитаете?

Сначала пользователь вводит имя объекта вместе со списком связанных с ним разрешенных значений. Этот ввод похож на пару "объект - значение", с той лишь разницей, что правая часть равенства может содержать несколько имен значений. Например, в приведенной выше консультации молодого человека, ввод для добавления разрешенных значений к первому вопросу будет выглядеть так: ОБРАЗОВАНИЕ=МАССАЧУСЕТСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ, КУЛИНАРНЫЙ КОЛЛЕДЖ, ШКОЛА ВОДИТЕЛЕЙ ГРУЗОВИКОВ. В более прагматическом примере, набор разрешенных значений для объекта *ЭКГ* примет вид: ЭКГ=НОРМА, ПОДНЯТЫЙ ST, ОТРИЦАТЕЛЬНЫЙ ST; и для объекта *температура*, ТЕМПЕРАТУРА=НОРМАЛЬНАЯ, ПОВЫШЕННАЯ, ПОНИЖЕННАЯ.

Затем оператор вводит текст вопроса. К объекту *ЭКГ* добавляется вопрос "Как выглядит зубец ST на ЭКГ пациента?". К имени объекта *температура* может быть добавлен вопрос "Какая температура у пациента?".

Программные модули для вопросов и разрешенных значений

Модули, описанные в этой главе, позволят экспертной системе воспринять ввод оператора, содержащий имя объекта и список разрешенных значений, и подсоединить имена этих значений к сцепленному списку объекта. Кроме того, программа будет снабжена устройством добавления вопросов об определенных объектах и вывода существующих вопросов с соответствующими разрешенными значениями на экран дисплея.

FIND_WORD находит разрешенное значение в строке, содержащей имя объекта и список разрешенных значений.

ADD_LEGAL добавляет разрешенное значение к имени определенного объекта.

FIND_LEGAL находит конкретное разрешенное значение в списке разрешенных значений объекта.

MAKE_LEGALS воздействует на строку, содержащую имя объекта и перечень разрешенных значений для выделения каждого имени списка.

ADD_QUESTION запоминает введенный оператором вопрос об определенном объекте.

P_QUESTION выводит на экран существующий вопрос о названном объекте.

ASK позволяет оператору ответить на вопрос, поставленный программой.

Определение расположения разрешенного значения в строке

Первый модуль просто будет определять расположение n-го значения в строке вида ОБЪЕКТ=ЗНАЧЕНИЕ1, ЗНАЧЕНИЕ2, ЗНАЧЕНИЕ3, ЗНАЧЕНИЕ4 и запоминать найденное значение в переменной WORD. Если n-е значение не найдено, функция принимает значение FALSE.

Эта функция сначала вычисляет номер имени разрешенного значения, содержащегося в строке, затем отмечает позицию следующей запятой в списке разрешенных значений. Если число значений меньше n, программа выдает значение

FALSE. Найдя n-е разрешенное значение, программа выдает TRUE и копирует имя значения в строковую переменную WORD.

```
FUNCTION find_word(f_line:line_string;n:integer;VAR
word:word_string):boolean;
VAR
x,
com_place:integer;
BEGIN
find_word:=FALSE;
word:='';
FOR x:=1 TO n DO
        BEGIN
        com_place:=pos(COMMA,f_line);
        IF (com_place=0) THEN
                BEGIN
                com_place:=length(f_line)+1;
                find_word:=TRUE
                END;
        word:=copy(f_line,1,com_place-1);
        f_line:=copy(f_line,com_place+1,length(f_line)-com_place)
        END
END;
```

Добавление узла к списку разрешенных значений

Этот модуль добавляет новый узел в вершину списка разрешенных значений объекта. Сначала процедура создает новый узел для имени значения. Затем она переопределяет вершину списка и устанавливает указатель *current_object^.legal_list* на этот новый узел. И наконец, она заносит в новый узел ссылку на вершину списка.

```
PROCEDURE add_legal(f_object:word_string;curr_object:object_ptr);

VAR
curr_value,
head:legal_ptr;
```

```
BEGIN
new(curr_value);
curr_value^.next:=NIL;
curr_value^.name:=f_object;
head:=curr_object^.legal_list;

IF (head<>NIL) THEN
        BEGIN
        WHILE (head^.next<>NIL) DO
                head:=head^.next;
        head^.next:=curr_value
        END
ELSE
        curr_object^.legal_list:=curr_value
END;
```

Вызов значений

Следующий модуль будет применяться для извлечения имени значения после нахождения разрешенного значения, соответствующего переменной *num* в списке резрешенных значений объекта.

Объявление переменных определяет указатели, которые отслеживают список объектов и список резрешенных значений. Кроме того, создается счетчик, принимающий целые значения. Четвертая объявленная величина определяет строковую переменную WORD. В нее заносится разрешенное значение, используемое в первом модуле.

Функция FIND_LEGAL сначала расщепляет строку, содержащую имя объекта и список разрешенных значений. Затем выполняется проверка наличия объекта в списке объектов. Если имя объекта найдено, функция принимает значение TRUE. Далее программа устанавливает значение счетчика на единицу и просматривает список разрешенных значений объекта до тех пор, пока не обнаружит заданный элемент. Если конец списка достигается прежде, чем найдено определенное разрешенное значение, выдается FALSE.

```
FUNCTION find_legal(f_object:word_string;n:integer;VAR
word:word_string)
        :boolean;

VAR
curr_object:object_ptr;
curr_value:legal_ptr;
counter:integer;

BEGIN
curr_object:=find_object(f_object);
find_legal:=TRUE;
IF (curr_object<>nil) THEN
        BEGIN
        curr_value:=curr_object^.legal_list;
        word:=curr_value^.name;
        counter:=1;
        IF (curr_value=NIL) THEN find_legal:=FALSE;
        WHILE ((curr_value<>NIL) AND (counter<n)) DO
                BEGIN
                curr_value:=curr_value^.next;
                IF (curr_value<>NIL) THEN
                        BEGIN
                        word:=curr_value^.name;
                        counter:=counter+1
                        END
                ELSE find_legal:=FALSE;
                END
        END
ELSE find_legal:=FALSE;
END;
```

Добавление разрешенных значений

Теперь, когда программа может извлекать из строки имена разрешенных значений, создавать узлы для новых разрешенных значений и находить разрешенное значение в списке, пора снабдить ее механизмом для запоминания добавленных разрешенных значений. Процедура MAKE_LEGALS воздействует на строку вида ОБЪЕКТ='ЗНАЧЕНИЕ1,ЗНАЧЕНИЕ2,ЗНАЧЕНИЕ3,ЗНАЧЕНИЕ4, отделяя каждое зна-

чение в строке и добавляя его к списку разрешенных значений объекта.

Эта процедура сначала расщепляет введенную строку для разделения имени объекта и имен разрешенных значений. Затем просматривается список объектов в поисках определенного объекта. Если имя объекта не найдено, программа вызывает процедуру MAKE_NODE для того, чтобы создать новый узел, а потом помещает новое имя объекта в вершину списка. Имена значений одно за другим добавляются к списку разрешенных значений объекта, пока не будет обработано последнее разрешенное значение в строке. При каждом добавлении значение счетчика увеличивается на единицу.

```
PROCEDURE make_legals(f_line:line_string);

VAR
curr_object:object_ptr;
counter,
st_place:integer;
new_line:line_string;
word:word_string;
done:boolean;
f_object,
dummy:word_string;

BEGIN
split(f_line,f_object,dummy);
curr_object:=find_object(f_object);
IF (curr_object=NIL) THEN make_node(curr_object);
curr_object^.name:=f_object;
st_place:=pos(EQUALS,f_line);
new_line:=copy(f_line,st_place+1,length(f_line)-st_place);
counter:=1;
done:=FALSE;
WHILE (done=FALSE) DO
        BEGIN
        done:=find_word(new_line,counter,word);
        add_legal(word,curr_object);
        counter:=counter+1;
        END
END;
```

Добавление вопросов

Модули, описываемые до сих пор в этой главе, относились к методам управления разрешенными значениями. Следующая процедура позволит оператору добавлять вопросы, основанные на разрешенных значениях, введенных ранее в базу знаний. ADD_QUESTION воздействует на введенную строку вида (ОБЪЕКТ)=(ВОПРОС), извлекая вопрос и помещая его в сцепленный список объектов.

Сначала процедура расщепляет строку, отделяя имя объекта от вопроса. Затем она ищет имя объекта в списке объектов. Если объект не найден, создается новый узел и данный объект помещается в вершину списка. Далее программа добавляет к объекту вопрос.

```
PROCEDURE add_question(f_line:line_string);

VAR
new_line:line_string;
curr_object:object_ptr;
f_object,
dummy:word_string;
st_place:integer;

BEGIN
split(f_line,f_object,dummy);
curr_object:=find_object(f_object);
IF (curr_object=NIL) THEN make_node(curr_object);
curr_object^.name:=f_object;
st_place:=pos(EQUALS,f_line);
new_line:=copy(f_line,st_place+1,length(f_line)-st_place);
curr_object^.question:=new_line
END;
```

Вывод вопроса на экран

Следующая процедура выводит на экран текст вопроса, задаваемого программой оператору. Если текст вопроса не задан, он будет формироваться автоматически в виде "Каково значение (ОБЪЕКТ)?".

```
PROCEDURE p_question(f_object:word_string);

VAR
curr_object:object_ptr;

BEGIN
curr_object:=find_object(f_object);

IF (curr_object<>NIL) THEN
        BEGIN
        IF (curr_object^.question<>'') THEN
        writeln(curr_object^.question)
        ELSE writeln('Каково значение ',f_object,'?')
        END
ELSE writeln('Каково значение ',f_object,'?')
END;
```

Ответ на вопрос

Следующая процедура позволяет оператору ответить на вопрос, заданный программой на основе разрешенных значений и предварительно введенного текста. Модуль ASK распознает введенное имя объекта и извлекает соответствующий текст вопроса. Из списка разрешенных значений объекта формируется меню, и программа ожидает, пока пользователь сделает выбор.

```
PROCEDURE ask(f_object:word_string;VAR f_value:word_string);

VAR
pick,
pick1,
num_vals:integer;
okay:boolean;
word,
select:word_string;

BEGIN
p_question(f_object);
```

```
IF (find_legal(f_object,1,word)=FALSE) THEN
        readln(f_value)
ELSE
        BEGIN
        num_vals:=1;
        WHILE (find_legal(f_object,num_vals,word)<>FALSE) DO
                BEGIN
                writeln(num_vals,'. ',word);
                num_vals:=num_vals+1
                END;
        pick:=0;
        WHILE ((pick<1) OR (pick>=num_vals)) DO
                BEGIN
                writeln('Пожалуйста введите номер от 1 до',
                num_vals-1);
                readln(select);
                pick:=ord(select[1])-48;
                IF (length(select)>1) THEN
                        BEGIN
                        pick1:=ord(select[2])-48;
                        IF ((pick1>=0) AND (pick1<10)) THEN
                        pick:=pick*10+pick1
                        END
                END;
        okay:=find_legal(f_object,pick,word);
        f_value:=word
        END
END;
```

Изменение основной программы

Для подключения новых модулей нам нужно изменить основную программу, добавив в меню возможности ввода разрешенных значений, добавления вопросов и вывода их на экран. Мы должны также увеличить число возможностей выбора с 5 до 8 и добавить шаги, на которых будут вызываться наши новые процедуры.

Кроме того, мы добавляем строки, описывающие новые возможности меню:

```
        writeln('5. Ввод разрешенных значений');
        writeln('6. Добавление вопросов об объекте');
        writeln('7. Ответ на вопрос об объекте');
```

Выбор "Выход" должен быть помечен цифрой 8. Ниже приведен полный листинг основной программы, включающий модули, введенные в этой главе.

```
BEGIN
max_choice:=7;
choice_lim:=max_choice+1;
last_try:=NIL;
top_fact:=NIL;
choice:=0;
WHILE (choice<>choice_lim) DO
        BEGIN
        writeln;
        writeln('1. Добавление факта в базу знаний');
        writeln('2. Проверка истинности факта');
        writeln('3. Просмотр фактов базы знаний');
        writeln('4. Объявление объекта многозначным');
        writeln('5. Ввод разрешенных значений');
        writeln('6. Добавление вопросов об объекте');
        writeln('7. Ответ на вопрос об объекте');
        writeln(choice_lim,'. Выход');
        writeln;
        write('Введите номер от 1 до ',max_choice,' и нажмите
        ENTER:');
        readln(choice);
        writeln;
        writeln;
        IF (choice>0) AND (choice<=choice_lim) THEN
        BEGIN
        CASE choice OF

                1:
                BEGIN
                writeln('Какой факт вы хотите добавить в базу
                знаний ?');
                writeln('(Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ),кл=
                (ЦЕЛОЕ))');
                readln(s_line);
```

```
s_cf:=get_cf(s_line);
split(s_line,s_object,s_value);
IF (ok_add(s_object,s_cf)=TRUE) THEN
        BEGIN
        add_object(s_object,s_value);
        add_cf(s_object,s_value,s_cf);
        writeln('Факт добавлен')
        END
ELSE
        BEGIN
        write('Добавление не разрешено.Объект(');
        writeln(s_object,') не многозначный.');
        END
END;

2:
BEGIN
writeln('Какой факт вы хотите проверить ?');
writeln('Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ)');
readln(s_line);
split(s_line,s_object,s_value);
IF test(s_object,s_value)=NIL THEN
        writeln('НЕВЕРНО')
ELSE
        writeln('ВЕРНО')
END;

3: see_objects(TRUE);

4:
BEGIN
writeln('Какой объект вы хотите объявить
многозначным ?');
writeln('(Введите:(ОБЪЕКТ))');
readln(s_object);
make_multi(s_object)
END;

5:
BEGIN
write('Введите разрешенное значение для
объекта:');
```

```
            writeln(' (Введите: (ОБЪЕКТ)=(СПИСОК))');
            readln(s_line);
            make_legals(s_line)
            END;

            6:
            BEGIN
            writeln('Какой вопрос вы хотите добавить ?');
            writeln('(Введите:(ОБЪЕКТ)=(ТЕКСТ))');
            readln(s_line);
            add_question(s_line);
            END;

            7:
            BEGIN
            writeln('Вопрос о каком факте');
            writeln('вы хотите просмотреть?');
            writeln('(Введите:(ОБЪЕКТ))');
            readln(s_object);
            ask(s_object,s_value);
            add_object(s_object,s_value);
            add_cf(s_object,s_value,DEFINITE)
            END

            END
        END
            ELSE writeln('Вы должны ввести число
            от 1 до',choice_lim)
        END
END.
```

Пробный запуск программы

Запустив новую основную программу на выполнение, вы сможете сначала добавить объекту список разрешенных значений, а затем - вопрос. Возможные ответы на этот вопрос содержатся в списке разрешенных значений, присоединенном к объекту. Чтобы увидеть систему в действии, выберите из меню возможность "Ответ на вопрос об объекте" и введите имя объекта.

Для проверки новых модулей мы доработаем нашу систему оценки состояния здоровья, использовавшуюся в

предыдущих главах. В гл. 6 мы ввели в базу знаний факты в виде пар "объект - значение", в гл. 7 снабдили выражения фактов коэффициентами доверия, в гл. 8 сделали объекты многозначными. В данной главе воспользуемся этими фактами для формирования вопросов и разрешенных значений, с помощью которых наша демонстрационная система будет проводить консультацию.

После загрузки программы выберите из меню возможность 5. Введите список разрешенных значений в следующем формате:

ОБЪЕКТ=ЗНАЧ1,ЗНАЧ2,...,ЗНАЧn

где n - последнее разрешенное значение. После ввода каждого списка выбирайте возможность 6 для описания вопроса. Здесь приведены разрешенные значения и вопросы, требующиеся нашей гипотетической программе оценки состояния здоровья.

Разрешенные значения

возраст=25_или_меньше,между_25_и_55,55_или_больше
вес=55_или_меньше,55-85,85_или_больше
рост=150_или_меньше,150-170,170_или_больше
телосложение=мелкое,крупное
курение=да,нет
потребление_холестерина=высокое,нормальное,низкое
потребление_растительных_жиров=высокое,нормальное,низкое
потребление_соли=высокое,нормальное,низкое
характер=агрессивный,мягкий
потребление_алкоголя=никакого,умеренное,чрезмерное
происхождение=сев_америка,средиземноморье
раса=европеоидная,негроидная,монголоидная

Вопросы

возраст=Сколько пациенту лет?
вес=Каков вес пациента?
рост=Каков рост пациента?
телосложение=Каково телосложение пациента?
курение=Курит ли пациент?
потребление_холестерина=Насколько велико потребление пациентом животных жиров?
потребление_растительных_жиров=Насколько велико

потребление пациентом растительных жиров?
потребление_соли=Каково содержание соли в пище?
потребление_кальция=Каково содержание кальция в пище?
потребление_алкоголя=Как оценить потребление пациентом алкоголя?
характер=Какие из перечисленных черт лучше описывают характер пациента?
происхождение=Откуда родом пациент?
раса=Какова расовая принадлежность пациента?

ГЛАВА 10

Чтение правил

Теперь наша программа экспертной системы имеет возможность читать, запоминать и вызывать пары "объект - значение"; читать, запоминать и изменять коэффициенты доверия; определять объекты как многозначные; читать, записывать и вызывать разрешенные значения; читать, запоминать и выводить на экран вопросы о названных объектах, основанные на их разрешенных значениях. До этого момента мы уделяли основное внимание накоплению объектов и значений, а также различным элементам интерфейса пользователя. В этой главе мы начнем работать над машиной вывода.

Правила экспертной системы

Экспертная система обрабатывает символическое представление реальности с помощью эвристических правил, обычно с помощью метода *обратной цепочки*. В этом методе консультация начинается с определения конкретной *цели* или конечного результата. Вспомните, в прологе пленник начинает консультацию с цели "освобождение". Врач может попросить медицинскую экспертную систему "подтвердить диагноз", а предприниматель - спросить об "оптимальных инвестициях". Этот подход принципиально отличается от *прямой цепочки*, когда оператор начинает с определения симптома или проблемы, а не с цели. В нашем случае мы сосредоточим внимание на методе достижения поставленной цели с помощью обратной цепочки.

Правило состоит из двух частей: *предпосылки* и *заключения*. Как предпосылка, так и заключение являются фактами базы знаний, выраженными парами "объект - значение". В

нашей программе экспертной системы правила имеют следующий формат:

Если ПРЕДПОСЫЛКА то ЗАКЛЮЧЕНИЕ

Эта простая концепция именуется *модус поненс*, "метод убеждения". Его смысл в том, что если верна предпосылка, то верно и заключение. Если мы примем утверждение "Если А, то Б", то мы также примем, что если А истинно, то истинно и Б.

Выражение правил

Таблица 10.1 иллюстрирует логику правил "если-то". В каждом случае мы принимаем факт, что если верна предпосылка, то верно и заключение. Поскольку заключение может быть не вполне определенным, пара "объект - значение" в части правила, представляющей заключение, снабжена коэффициентом доверия.

Таблица. 10.1.

Иллюстрация правил "если - то"

Предпосылка	Заключение
ЕСЛИ ЭКГ=отрицательный_ST	ТО миокард=ишемичен
ЕСЛИ цена=статична	ТО контракт=фиксированная_цена
ЕСЛИ температура=высокая	ТО лихорадка=да
ЕСЛИ занятие=библиотекарь	ТО интересы=литература

Некоторые из правил экспертной системы, описанные в прологе, могли бы выглядеть так:

```
правило1:    если
                 солнечный_свет=прямой
то
                 направление=юг.
правило2:    если
                 солнечный_свет=не_прямой
то
                 направление=север,кд=100.
```

Эти простые отношения "если-то" представляют узлы решения, по которым машина вывода продвигается к поставленной цели. В качестве более практического примера возьмем случай с покупателем, сравнивающим возможные методы получения скидки. Допустим, пользователь рассматривает две возможности: прямую (цена покупки минус страховка, деленная на срок службы), или УВС - ускоренное возмещение стоимости (фиксированный процент от цены покупки, зависящий от типа имущества). В число характеристик, учитываемых соответствующим методом, входят: тип имущества, процент пользы, приносимой бизнесу, и цена приобретения. На основе законов о налогах 1985 г. некоторые правила налоговой экспертной системы могли бы выглядеть так:

```
правило29:   если
             имущество=автомобиль
то
             тип_УВС=3-годичное_имущество.
правило30:   если
             имущество=передвижной_дом
то
             тип_УВС=10-летнее_имущество
правило31:   если
             тип_УВС=3-годичное имущество
то
             годовая_скидка=25_процентов
```

В приведенном примере коэффициент определенности в каждом случае равен 100, так как эти правила весьма точны. Налоговые законы весьма определенны, а наибольшую определенность имеет смерть.

Правило может включать в себя булевский оператор "и" для образования таких выражений как "Если А и если Б, то В", так как не все решения линейны. Следовательно, экспертная система может обрабатывать правила такого вида:

```
правило50:   если
             изменение_дохода=140%+_возрастание и
             метод_прошлого_года=не_усреднение
то
             рекомендуемый_метод=усреднение.
```

правило51: если
проценты_по_закладной=$2400 и
медицинские_расходы=доходХ5%
то
рекомендуемый_метод=
вычеты_по_статьям.

Для ввода правил в экспертную систему пользователь сначала записывает их в текстовый файл с помощью любого текстового редактора (*edlin* MS-DOS, *ed* CP/M) или программы подготовки текстов (WordStar, IBM Writing Assistant или MultiMate). Затем программа считывает этот текстовый файл и включает правила в свою базу знаний.

Программные модули для чтения, сохранения и извлечения правил

В этой главе мы введем модули, которые позволят нашей программе экспертной системы читать, сохранять и извлекать правила, аналогичные приведенным в предыдущих примерах.

P_READ будет выполнять некоторые простые вспомогательные операции, такие как преобразование букв вводимой строки в строчные и удаление пробелов.

ADD_PREM будет использоваться для добавления к текущему правилу предпосылки.

ADD_CON будет добавлять соответствующее заключение.

P_RULE обрабатывает указатель правила для вывода названного правила на экран дисплея.

ENTER_RULE читает правило из текстового файла и заносит его в список правил экспертной системы.

READ_FILE будет использоваться для чтения всего содержимого файла правил в базу знаний.

Вспомогательные операции

Для облегчения ввода эта процедура будет упаковывать строки в стандартный формат, исключая пробелы и преобразуя все буквы в строчные. Пробелы, стоящие между словами в вопросе будут оставлены для четкости и ясности.

```
PROCEDURE p_read(VAR oline:line_string);
VAR
c:char;
cs:string[1];
len,
counter,
st_place:integer;
supress:boolean;
in_line:line_string;

BEGIN
readln(rules,in_line);
oline:='';
len:=length(in_line);
st_place:=pos(' и',in_line); (Обратите внимание на пробел после
                                                  первого апострофа.)
IF (st_place>0) THEN len:=st_place;
supress:= FALSE;
FOR counter:=1 TO len DO
        BEGIN
        c:=in_line[counter];
        IF ((c=EQUALS) AND (pos('вопрос',oline)>0)) THEN
                                        supress:=TRUE;
        IF (ord(c)=9) THEN c:=' ';  (Обратите внимание на пробел
                                        между апострофами.)
        IF ((c in['А'..'П']) AND (supress=FALSE)) THEN
                c:=chr(ord(c)+32);*
        IF ((c in['Р'..'Я']) AND (supress=FALSE)) THEN
                c:=chr(ord(c)+80);
        cs:=' '; (Обратите внимание на пробел между
                        апострофами.)
        cs[1]:=c;
        IF((c<>' ') OR (supress=TRUE)) THEN
                oline:=concat(oline,cs)
        END
END;
```

---

* Приведенная здесь программа рассчитана на использование для кодирования кириллицы альтернативного набора ГОСТа. При использовании основного набора операторы перекодировки следует соответствующим образом изменить. — *Примеч. пер.*

Добавление предпосылок

Теперь перейдем к добавлению предпосылок и заключений, которые составляют правила экспертной системы. Приведенная ниже функция добавляет к текущему правилу предпосылку. Каждое правило имеет список предпосылок, содержащий предпосылки этого правила, и список заключений. ADD_PREM начинается с операторов, расщепляющих исходную строку, для извлечения предпосылки, выраженной парой "объект - значение". Затем программа создает новый узел списка предпосылок правила, устанавливая указатель правила *curr_rule* на соответствующее имя объекта и имя значения. Завершает подпрограмму добавление к списку нового узла.

```
FUNCTION
add_prem(curr_prem:prem_ptr;f_line:line_string):prem_ptr;
VAR
temp,
new_prem:prem_ptr;
f_object,
f_value:word_string;

BEGIN
split(f_line,f_object,f_value);
add_prem:=curr_prem;
new(new_prem);
WITH new_prem^ DO
        BEGIN
        object:=f_object;
        value:=f_value;
        next:=NIL;
        END;
IF (curr_prem=NIL) THEN add_prem:=new_prem
ELSE
        BEGIN
        WHILE (curr_prem^.next<>NIL) DO
                curr_prem:=curr_prem^.next;
        curr_prem^.next:=new_prem
        END
END;
```

Добавление заключения

С помощью этой функции к текущему правилу добавляется заключение. ADD_CON расщепляет вводимую строку, выделяя заключение в форме пары "объект - значение". Затем программа создает новый узел в вершине списка заключений правила и добавляет в него новое заключение.

```
FUNCTION add_con(curr_con:con_ptr;f_line:line_string):con_ptr;
VAR
temp,
new_con:con_ptr;
f_object,
f_value:word_string;

BEGIN
split(f_line,f_object,f_value);
add_con:=curr_con;
new(new_con);
WITH new_con^ DO
        BEGIN
        object:=f_object;
        value:=f_value;
        cert:=get_cf(f_line);
        next:=NIL
        END;
IF (curr_con=NIL) THEN add_con:=new_con
ELSE
        BEGIN
        WHILE (curr_con^.next<>NIL) DO
                curr_con:=curr_con^.next;
        curr_con^.next:=new_con
        END
END;
```

Вывод правил на экран

Этот модуль позволяет программе по запросу выводить существующее правило на экран. P_RULE обрабатывает указатель правила, находит правило и выводит на экран дисплея его текст.

```
PROCEDURE p_rule(curr_rule:rule_ptr);

VAR
curr_prem:prem_ptr;
curr_con:con_ptr;
BEGIN
writeln(curr_rule^.name,': если');
curr_prem:=curr_rule^.prem;

WHILE (curr_prem<>NIL) DO
        BEGIN
        write(curr_prem^.object,'=');
        write(curr_prem^.value);
        curr_prem:=curr_prem^.next;
        IF (curr_prem<>NIL) THEN writeln( ' и ')
        ELSE writeln
        END;
writeln(' то ');
curr_con:=curr_rule^.con;

WHILE (curr_con<>NIL) DO
        BEGIN
        write(curr_con^.object,'=');
        write(curr_con^.value,',кл=',curr_con^.cert);
        curr_con:=curr_con^.next;
        IF (curr_con<>NIL) THEN writeln(' и ')
        ELSE writeln
        END
END;
```

Добавление правил в базу знаний

Следующая процедура позволяет программе считать из текстового файла одно правило и добавить его к списку правил. Компоненты правил - предпосылка и заключение - обрабатываются раздельно. Процедура ENTER_RULE начинается операторами, создающими новый узел в вершине списка правил. Затем программа считывает из текстового файла предпосылку и добавляет ее к списку. И наконец, она считывает заключение и помещает его в узел.

```
PROCEDURE enter_rule(rule_name:word_string);

VAR
new_rule,
curr_rule:rule_ptr;
line:line_string;
done:boolean;

BEGIN
new(new_rule);
IF (top_rule<>NIL) THEN
        BEGIN
        curr_rule:=top_rule;
        WHILE (curr_rule^.next<>NIL) DO
                curr_rule:=curr_rule^.next;
        curr_rule^.next:=new_rule
        END
ELSE top_rule:=new_rule;
WITH new_rule^ DO
        BEGIN
        name:=rule_name;
        next:=NIL;
        prem:=NIL;
        con:=NIL
        END;
p_read(line);
done:=FALSE;
WHILE ((NOT done) AND (NOT Eof(rules))) DO
        BEGIN
        new_rule^.prem:=add_prem(new_rule^.prem,line);
        p_read(line);
        IF (pos('TO',line)>0) AND (length(line)=2) THEN
                done:=TRUE
        END;
 p_read(line);
 done:=FALSE;
 REPEAT
        IF (Eof(rules)) THEN done:=TRUE;
        new_rule^.con:=add_con(new_rule^.con,line);
        IF line[length(line)]='.' THEN done:=TRUE
        ELSE p_read(line)
UNTIL(done);
```

```
writeln;
p_rule(new_rule)
END;
```

Чтение файла правил

Теперь нашей программе потребуется возможность считать весь файл правил в свою базу знаний. Следующая процедура, READ_FILE, читает файл, содержащий подготовленные пользователем правила, и обрабатывает встречающиеся команды для присвоения объекту статуса многозначности, добавления вопросов или добавления к имени объекта разрешенных значений. Затем она вызывает процедуру ENTER_RULE для занесения каждого правила в соответствующий список правил.

```
PROCEDURE read_file;

VAR
command:word_string;
m_line,
f_line:line_string;
st_place:integer;

BEGIN
writeln('Чтение файла, содержащего правила');
assign(rules,'rules');
reset(rules);
top_rule:=NIL;
command:='';
WHILE (NOT Eof(rules)) DO
        BEGIN
        p_read(f_line);
        st_place:=pos('(',f_line);
        IF (st_place=0) THEN st_place:=pos(COLON,f_line);
        IF (st_place>1) THEN
                BEGIN
                command:=copy(f_line,1,st_place-1);
                m_line:=copy(f_line,st_place+1,length(f_line)-
                        st_place);
                IF (command='многозначный') THEN
                        make_multi(m_line);
```

```
            IF (command='вопрос') THEN add_question(m_line);
            IF (command='разрешен') THEN
                    make_legals(m_line);
            IF (pos('правило',command)>0) THEN
                    enter_rule(command)
            END
    END
END;
```

Создание файла правил

Для создания файла правил можно использовать любой текстовый редактор, который создает файлы в коде ASCII. Файлу должно быть присвоено имя *rules*. При вводе каждого правила соблюдайте следующий синтаксис:

```
правило(n):   если
              ПРЕДПОСЫЛКА
то
              ЗАКЛЮЧЕНИЕ.
```

Проследите за тем, чтобы каждое правило заканчивалось точкой. Если точку опустить, программа не сможет правильно считать этот файл. Ниже приведены примеры верной записи правил.

```
правило9:     если
              курильщик=да
то
              риск_рака=выше_среднего.
правило21:    если
              характер=агрессивный
то
              тип_характера=тип_b.
правило35:    если
              возраст=35 и
              пол=м
то
              основная_продолжительность=70.
```

Хотя новые модули позволяют системе читать файл данных на текущем дисководе, они еще не являются механизмом применения правил для достижения поставленной цели. Этот механизм, который можно назвать сердцем или, точнее, мозгом экспертной системы, и очередной модуль мы рассмотрим в следующей главе.

ГЛАВА 11

Достижение цели

Теперь наша программа имеет возможность считывать как правила, так и факты. Для установления фактов в базе знаний экспертная система добавляет имена объектов к сцепленному списку. Каждый узел списка объектов имеет отдельный список значений, содержащий имена связанных с ним значений. Факт может быть однозначным или многозначным. Мы также создали инструментарий для определения разрешенных значений названного объекта и создания связанного с ним вопроса, что предоставляет оператору возможность выбора. В предыдущей главе мы обеспечили программе возможность чтения правил "если-то", устанавливающих отношения между фактами базы знаний.

В этой главе мы снабдим программу машиной вывода.

Машина вывода

База знаний представляет описание предметной области экспертной системы. Машина вывода является *интерпретатором правил*, который использует факты этой базы знаний для решения поставленных проблем. Она осуществляет это путем формулирования пробных гипотез и проверки их на соответствие указанной цели. Оператор задает цель консультации в виде имени объекта. Машина вывода использует набор правил, пытаясь получить значение указанного объекта-цели. Программа продолжает поиск до тех пор, пока одно из предполагаемых решений не окажется верным.

В системах, применяющих подобно нашей программе обратную цепочку, вывод начинается с конечного заключения, объекта-цели. На рис. 11.1 показан этот процесс. Программа перебирает список правил в поисках такого, которое содержит объект-цель в своей правой части. Затем машина проверяет

каждую часть предпосылки правила и процесс начинается снова.

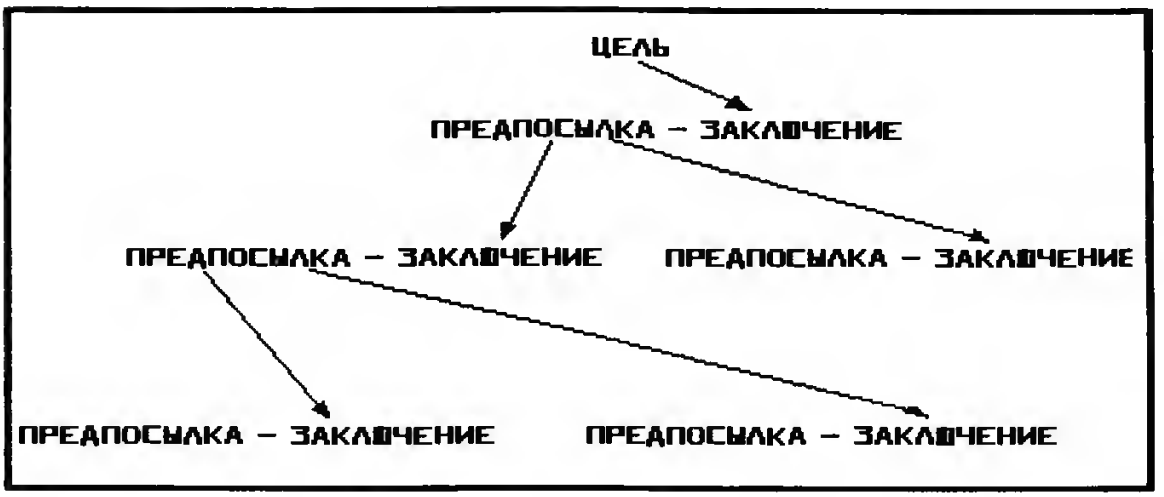


Рис. 11.1. Связь предпосылок и заключений

На рис. 11.2 мы видим, как машина вывода выполняет поиск в базе знаний для подтверждения предполагаемого конечного заключения. В данном примере цель - выдача рекомендаций по подходящей методике использования налоговой стратегии. Рассматриваемые правила касаются величины страховки и типа имущества, на которое определяется скидка. Иллюстрация на рис. 11.2 представляет несколько гипотетических правил.

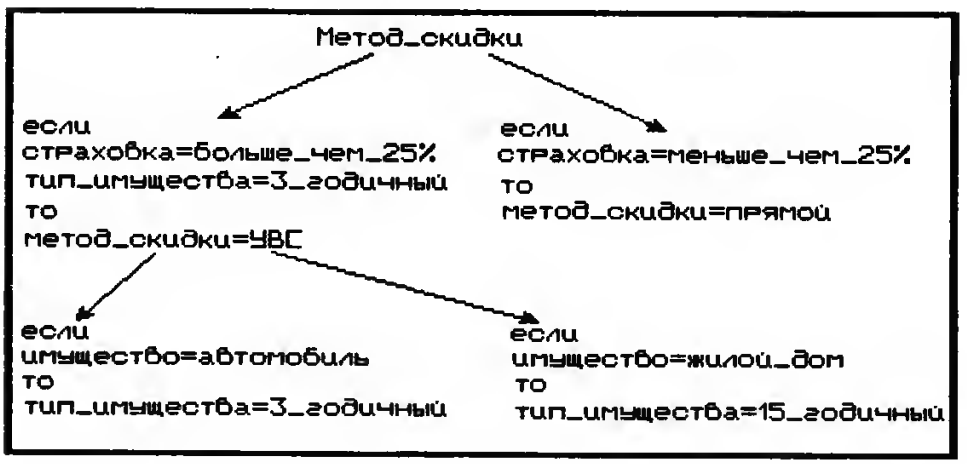


Рис. 11.2. Пример связи правил определения скидки

Работа начинается с поиска правила, содержащего в правой части объект-цель *метод_скидки*. В данном случае оказываются два таких правила, одно из которых рекомендует прямой метод, а другое - метод УВС. Затем программа строит обратную цепочку для проверки фактов каждой предпосылки.

Программа продвигается таким образом до тех пор пока не будут исчерпаны все подходящие правила.

Программные модули для машины вывода

В этой главе мы введем четыре новых модуля для анализа правил с помощью машины вывода.

FIND_RULE будет просматривать список правил в поисках первого правила, содержащего объект, значение которого устанавливается в данный момент.

CONCLUDE осуществляет заключения правила.

PURSUE обрабатывает имя объекта и пытается применить правила для присвоения объекту значения.

P_RESULT обеспечивает вывод на экран результатов консультации.

Поиск подходящего правила

Эта функция отыскивает правило, содержащее в правой части имя заданного объекта. Она применяет указатель заключения *curr_con* для просмотра списка правил до совпадения имени объекта с заданным. Если соответствующий объект найден, функция принимает значение TRUE.

```
FUNCTION find_rule(obj:word_string;curr_rule:rule_ptr):rule_ptr;

VAR
found:boolean;
curr_con:con_ptr;
BEGIN
found:=FALSE;
find_rule:=NIL;
WHILE ((curr_rule<>NIL) AND (found=FALSE)) DO
        BEGIN
        curr_con:=curr_rule^.con;
        WHILE (curr_con<>NIL) DO
```

```
                BEGIN
                IF (curr_con^.object=obj) THEN
                        BEGIN
                        found:=TRUE;
                        find_rule:=curr_rule
                        END;
                curr_con:=curr_con^.next
                END;
        curr_rule:=curr_rule^.next
        END
END;
```

Осуществление заключения

Следующий модуль представляет собой процедуру для осуществления заключения, когда машина вывода находит связь между предпосылкой одного правила и заключением другого. Процедура CONCLUDE на основе целых значений кд рассматриваемых правил устанавливает коэффициент доверия заключения и выводит заключение на экран дисплея.

```
PROCEDURE conclude(curr_rule:rule_ptr;prem_cert:integer);
VAR
curr_con:con_ptr;
cert:integer;

BEGIN
curr_con:=curr_rule^.con;
        WHILE (curr_con<>NIL) DO
        BEGIN
                add_object(curr_con^.object,curr_con^.value);
                cert:=(prem_cert*curr_con^.cert) DIV 100;
                add_cf(curr_con^.object,curr_con^.value,cert);
                curr_con:=curr_con^.next
        END
END;
```

Результирующие факты

Следующий модуль является сердцем машины вывода программы. Процедура PURSUE обрабатывает полученное имя объекта, пытаясь применять правила для присвоения ему

значения. Если ни одно из правил не позволяет сделать вывод об объекте, то его значение запрашивается у пользователя.

Для фиксации текущего правила, рассматриваемого в данный момент, процедура использует указатель правил. Работа начинается с просмотра списка объектов для проверки наличия искомого объекта. Если объект не найден, в вершине списка создается новый узел.

Затем программа ищет правило, содержащее в правой части имя этого объекта. Далее она проверяет каждую составляющую предпосылки правила. Если правило выполняется, его заключение добавляется в базу знаний. Если же значение объекта с помощью правил найти не удается, программа предлагает оператору ввести имя этого значения.

```
PROCEDURE pursue(f_object:word_string);

VAR
f_value:word_string;
curr_object:object_ptr;
curr_value:value_ptr;
curr_rule:rule_ptr;
curr_prem:prem_ptr;
bad:boolean;
solved:boolean;
lowest:integer;
BEGIN
curr_object:=find_object(f_object);
IF curr_object=NIL THEN make_node(curr_object);
curr_object^.name:=f_object;
IF (curr_object^.sought<>TRUE) THEN
        BEGIN
        solved:=FALSE;
        curr_object^.sought:=TRUE;
        curr_rule:=find_rule(f_object,top_rule);
        WHILE ((curr_rule<>NIL) AND
                (ok_add(f_object,DEFINITE)=TRUE)) DO
                BEGIN
                curr_prem:=curr_rule^.prem;
                bad:=FALSE;
                lowest:=DEFINITE;
                WHILE ((curr_prem<>NIL) AND (bad=FALSE)) DO
```

```
                    BEGIN
                    pursue(curr_prem^.object);
                    curr_value:=test(curr_prem^.object,
                    curr_prem^.value);
                    IF curr_value=NIL THEN bad:=TRUE
                    ELSE IF curr_value^.cert<lowest THEN
                            lowest:=curr_value^.cert;
                    curr_prem:=curr_prem^.next
                     END;
            IF (bad=FALSE) THEN
                    BEGIN
                    conclude(curr_rule,lowest);
                    solved:=TRUE
                    END;
            curr_rule:=find_rule(f_object,curr_rule^.next)
            END;
    IF (solved=FALSE) THEN
            BEGIN
            ask(f_object,f_value);
            add_object(f_object,f_value);
            add_cf(f_object,f_value,DEFINITE)
            END
    END
END;
```

Печать результатов консультации

Следующая подпрограмма позволяет системе выводить на экран результаты консультации. Процедуре P_RESULT из основной программы передается имя объекта-цели. Она распечатывает на экране дисплея список всех значений этого объекта.

```
PROCEDURE p_result(f_object:word_string);

VAR
curr_object:object_ptr;

BEGIN
writeln;
writeln('РЕЗУЛЬТАТ КОНСУЛЬТАЦИИ:');
writeln;
```

```
curr_object:=find_object(f_object);
see_vals(curr_object,TRUE);
writeln;
writeln('КОНЕЦ КОНСУЛЬТАЦИИ.')
END;
```

Изменение основной программы

Изменив основную программу для включения новых модулей, ее можно запустить в действие, так как к настоящему моменту у нас есть достаточно модулей для проведения консультации.

В приведенном ниже листинге дана вся основная программа, включая модификации для вызова новых подпрограмм. В первую очередь, это введение новой возможности для оператора - "Достижение цели". Когда в качестве выбора вводится 8, программа предлагает оператору ввести цель в форме имени объекта. Затем она считывает это имя и вызывает процедуру *pursue*.

```
BEGIN
max_choice:=8;
choice_lim:=max_choice+1;
last_try:=NIL;
top_fact:=NIL;
choice:=0;
read_file;
WHILE (choice<>choice_lim) DO
        BEGIN
        writeln;
        writeln('1. Добавление факта в базу знаний');
        writeln('2. Проверка истинности факта');
        writeln('3. Просмотр фактов базы знаний');
        writeln('4. Объявление объекта многозначным');
        writeln('5. Ввод разрешенных значений');
        writeln('6. Добавление вопросов об объекте');
        writeln('7. Ответ на вопрос об объекте');
        writeln('8. Достижение цели');
        writeln(choice_lim,'. Выход');
        writeln;
        write('Введите номер от 1 до ',max_choice,' и нажмите
                ENTER:');
```

```
readln(choice);
writeln;
writeln;

IF (choice>0) AND (choice<=choice_lim) THEN
        BEGIN
        CASE choice OF
                1:
                BEGIN
                writeln('Какой факт вы хотите добавить в
                        базу знаний ?');
                writeln('(Введите:(ОБЪЕКТ)=(ЗНАЧЕНИЕ)
                        ,кд=(ЦЕЛОЕ))');
                readln(s_line);
                s_cf:=get_cf(s_line);
                split(s_line,s_object,s_value);
                IF (ok_add(s_object,s_cf)=TRUE) THEN
                        BEGIN
                        add_object(s_object,s_value);
                        add_cf(s_object,s_value,s_cf);
                        writeln('Факт добавлен')
                        END
                ELSE
                        BEGIN
                        write('Добавление не разрешено.
                                Объект  ('s_object,'));
                        writeln( не объявлен
                                многозначным.');
                        END
                END;

                2:
                BEGIN
                writeln('Какой факт вы хотите
                        проверить ?');
                writeln('Введите:(ОБЪЕКТ)=
                        (ЗНАЧЕНИЕ)');
                readln(s_line);
                split(s_line,s_object,s_value);
                IF test(s_object,s_value)=NIL THEN
                        writeln('НЕВЕРНО')
```

```
ELSE
        writeln('ВЕРНО')
END;

3: see_objects(TRUE);

4:
BEGIN
writeln('Какой объект вы хотите
объявить многозначным ?');
writeln('(Введите:(ОБЪЕКТ))');
readln(s_object);
make_multi(s_object)
END;

5:
BEGIN
write('Введите разрешенное значение для
        объекта:');
writeln(' (Введите:
        (ОБЪЕКТ)=(СПИСОК))');
readln(s_line);
make_legals(s_line)
END;

6:
BEGIN
writeln('Какой вопрос вы хотите
        добавить ?');
writeln('(Введите:(ОБЪЕКТ)=(ТЕКСТ))');
readln(s_line);
add_question(s_line);
END;

7:
BEGIN
writeln('О чем вы хотите получить
         вопрос ?');
writeln('(Введите:(ОБЪЕКТ))');
readln(s_object);
ask(s_object,s_value);
add_object(s_object,s_value);
```

```
                    add_cf(s_object,s_value,DEFINITE)
                    END;

                    8:
                    BEGIN
                    writeln('Какова ваша цель ?');
                    writeln('(Введите:(ОБЪЕКТ))');
                    readln(s_object);
                    pursue(s_object);
                    p_result(s_object)
                    END

                    END
            END
        ELSE writeln('Вы должны ввести число от 1
                ло',choice_lim)
        END
END.
```

Испытание системы

Хотя программа нашей экспертной системы теперь содержит достаточно команд для проведения настоящей консультации, ей нужна база знаний. Чтобы испытать программу, сначала скомпилируйте новые модули и переработайте основную программу. Затем сделайте следующее:

1. Используя текстовый редактор (например, редактор PC DOS *edlin*, или программу типа WordStar), создайте текстовый файл, содержащий ряд правил с парами "объект - значение" в предпосылке и заключении. Для ввода правил воспользуйтесь следующим форматом:

```
правило1:    если
             [ОБЪЕКТ]=[ЗНАЧЕНИЕ1]
        то
             [ОБЪЕКТ]=[ЗНАЧЕНИЕ2].
```

При чтении файла правил, программа ищет точку в конце каждого правила. Если вы опустите эту точку, система не сможет отличить одно правило от другого.

Вы можете также добавить в этот текстовый файл предложения "вопрос", "разрешен" и "многозначный".

2. Запустите основную программу.

3. Для проведения консультации выберите возможность "Достижение цели".

4. Введите имя объекта, представляющего цель, и нажмите клавишу RETURN.

Пробный запуск программы

При запуске программы *expert*, она считает правила из текстового файла на текущий дисковод, введет их в базу знаний, а затем выведет на экран меню возможностей оператора.

Кроме правил, текстовый файл может содержать предложения с разрешенными значениями, вопросами и множественными значениями. Когда программа читает файл правил, она добавляет эти элементы к базе знаний, избавляя вас от необходимости вводить их с помощью программы меню. В следующей главе мы рассмотрим метод ввода в файл правил полной базы знаний.

РАЗДЕЛ 3

ЭКСПЕРТНАЯ СИСТЕМА В ДЕЙСТВИИ

ГЛАВА 12

Создание подсистемы объяснения

Теперь программа нашей экспертной системы достаточно разработана, чтобы проводить консультацию. Она может считывать из текстового файла факты, вопросы, разрешенные значения и правила и затем следовать к цели на основе предпосылок и заключений введенных правил. В настоящей главе мы введем программные модули, позволяющие пользователю следить за логикой системы. Новые процедуры помогут экспертной системе "объяснить" пользователю ход ее рассуждений. Точнее, она будет показывать, как было получено конкретное заключение, и почему для достижения поставленной цели должен быть задан определенный вопрос.

Способность программы "пояснять" свои рассуждения является одной из необходимых характеристик хорошей экспертной системы. Помимо демонстрации подхода программы к решению проблемы, эта подсистема позволяет оператору контролировать обработку правил, созданных пользователем.

Допустим, что вы консультируетесь с экспертной системой, описанной в прологе. При использовании уже созданных модулей процесс взаимодействия может выглядеть примерно так:

Какова ваша цель?

(Введите: (ОБЪЕКТ))

путь_к_выходу

Какого цвета стены в вашей комнате?
1. серые
2.черные
2

Заключение.....местонахождение=ваша_комната кд 90

Какого цвета стены в коридоре?
1. серые
2. черные
1

Заключение.....уровень=второй_этаж кд 100

Какой свет попадает через иллюминатор?
1. прямой
2. непрямой

Допустим, вы хотите, чтобы в этом месте система приостановила работу и выдала объяснение. Если в основной программе появится строка EXPLAIN:=TRUE, то консультация будет проходить следующим образом:

Так как в базе знаний отсутствует правило для вывода значения объекта свет
то надо запросить пользователя.

Какой свет попадает через иллюминатор?
1. прямой
2. непрямой

Система поясняет, что она не может дальше следовать к цели, не установив значение объекта "свет". Как только пользователь введет недостающее значение, программа продолжит проверку предпосылок и заключений, пока не потребуется другое значение или не будет достигнуто решение проблемы.

Давайте рассмотрим другой пример. Допустим, вы консультируетесь с экспертной системой для определения наилучшего метода скидки при анализе очередной выплаты годового налога. Консультация с нашей демонстрационной программой может выглядеть примерно так:

```
Какова ваша цель?
метод_скидки

Какова закупочная стоимость имущества?
1. менее $1000
2. между $1000 и $5000
3. свыше $5000
2

Каков вид собственности?
1. моторное средство передвижения
2. недвижимость
3. торговое оборудование
1

Как часто это средство используется
в деловых целях?
1. менее 50% времени
2. 50% времени или больше
1

Заключение.....имущество=без_скидки
```

Система пришла к заключению, что этот вид собственности не дает права на скидку при налогообложении. Допустим, вы хотите получить объяснение, каким образом система сделала такой вывод.

```
Как часто это средство используется
в деловых целях?
1. менее 50% времени
2. 50% времени или больше
1
```

Заключение.....имущество=без_скидки

Так как: если использование_средства=менее_50%
можно сделать вывод, что имущество=без_скидки

В этом примере система в качестве объяснения выводит правило. Ответ имеет две составляющие: первая - это текущая предпосылка, а вторая - результирующее заключение.

Новые модули

В этой главе мы введем две новые процедуры.

EXPLAIN_HOW позволяет программе "объяснить" ход своих рассуждений, выводя на экран правило или правила, которыми она воспользовалась для получения конкретного заключения.

EXPLAIN_WHY обеспечивает системе возможность отвечать, почему должен быть задан тот или иной вопрос.

Эти новые процедуры будут использоваться программой только в том случае, когда в основной программе появится строка EXPLAIN=TRUE.

Кроме того, мы модифицируем введенную в гл. 10 процедуру PURSUE для включения этих новых возможностей.

Объяснение вывода

Следующий модуль позволяет "объяснить" вывод в ходе консультации. При вызове процедуры EXPLAIN_HOW, на экран выводится надпись "Так как:", затем - содержание текущей предпосылки в формате ОБЪЕКТ=ЗНАЧЕНИЕ. Если в выводе участвует более одной предпосылки, программа продолжает выводить их на экран до тех пор, пока указатель текущей предпосылки не примет значение *nil*.

В конце объяснения программа пишет: "Можно сделать вывод, что" и выдает текущее заключение (тоже в формате ОБЪЕКТ=ЗНАЧЕНИЕ). Если получено несколько заключений, программа продолжает выводить их до тех пор, пока не будут исчерпаны все заключения.

```
PROCEDURE explain_how(curr_rule:rule_ptr);

VAR
curr_prem:prem_ptr;
curr_con:con_ptr;
BEGIN
writeln;
writeln('Так как :');
curr_prem:=curr_rule^.prem;
WHILE (curr_prem<>NIL) DO
        BEGIN
        write(curr_prem^.object,'=');
        write(curr_prem^.value);
        curr_prem:=curr_prem^.next;
        IF (curr_prem<>NIL) THEN writeln(' и ')
        ELSE writeln
        END;

writeln(' Можно сделать вывод, что');
curr_con:=curr_rule^.con;

WHILE (curr_con<>NIL) DO
        BEGIN
        write(curr_con^.object,'=');
        write(curr_con^.value,', к.опр.=',curr_con^.cert);
        curr_con:=curr_con^.next;
        IF (curr_con<>NIL) THEN writeln(' и ')
        ELSE writeln
        END;
writeln
END;
```

Объяснение, почему был задан вопрос

Кроме объяснения, как был сделан вывод, программа должна иметь возможность пояснять, почему был задан вопрос. Следующая процедура, EXPLAIN_WHY, вызывается, когда система приходит к какому-либо заключению, но при этом в основной программе должна присутствовать строка EXPLAIN=TRUE.

```
PROCEDURE explain_why(f_object:word_string);

BEGIN
writeln;
writeln('Так как в БЗ отсутствует правило');
writeln('вывода значения для объекта ',f_object);
writeln('то нужно запросить пользователя.');
writeln
END;
```

Изменения в программе

Для того чтобы включить в программу новые модули, нам нужно добавить в процедуру PURSUE две строки: одну для вызова EXPLAIN_HOW и одну для вызова процедуры EXPLAIN_WHY. Кроме того, в основную программу перед первым оператором WHILE нужно добавить предложение EXPLAIN:=TRUE. Процедуры EXPLAIN_HOW и EXPLAIN_WHY нужно вставить в программу непосредственно перед процедурой PURSUE.

Ниже приведен полный листинг подпрограммы PURSUE с новыми предложениями. Переработанная основная программа целиком приведена в Приложении А.

```
PROCEDURE pursue(f_object:word_string);

VAR
f_value:word_string;
curr_object:object_ptr;
curr_value:value_ptr;
curr_rule:rule_ptr;
curr_prem:prem_ptr;
bad:boolean;
solved:boolean;
lowest:integer;

BEGIN
curr_object:=find_object(f_object);
IF curr_object=NIL THEN make_node(curr_object);
curr_object^.name:=f_object;
IF (curr_object^.sought<>TRUE) THEN
```

```
        BEGIN
        solved:=FALSE;
        curr_object^.sought:=TRUE;
        curr_rule:=find_rule(f_object,top_rule);
        WHILE ((curr_rule<>NIL) AND
                (ok_add(f_object,DEFINITE)=TRUE)) DO
                BEGIN
                curr_prem:=curr_rule^.prem;
                bad:=FALSE;
                lowest:=DEFINITE;
                WHILE ((curr_prem<>NIL) AND (bad=FALSE)) DO
                        BEGIN
                        pursue(curr_prem^.object);
                        curr_value:=test(curr_prem^.object,
                                curr_prem^.value);
                        IF curr_value=NIL THEN bad:=TRUE
                        ELSE IF curr_value^.cert<lowest THEN
                                lowest:=curr_value^.cert;
                        curr_prem:=curr_prem^.next
                        END;
                IF (bad=FALSE) THEN
                        BEGIN
                        IF (explain=TRUE) THEN
                                explain_how(curr_rule);
                        conclude(curr_rule,lowest);
                        solved:=TRUE
                        END;
                curr_rule:=find_rule(f_object,curr_rule^.next)
                END;
        IF (solved=FALSE) THEN
                BEGIN
                IF (explain=TRUE) THEN explain_why(f_object);
                ask(f_object,f_value);
                add_object(f_object,f_value);
                add_cf(f_object,f_value,DEFINITE)
                END
        END
END;
```

ГЛАВА 13

Использование экспертных систем

Для того чтобы задействовать нашу экспертную систему, нам нужно создать осмысленную базу знаний для конкретной предметной области. Как было показано в гл. 3, этот процесс требует определения целей трех типов: конечных, промежуточных и вспомогательных.

Постановка конечных целей

Прежде всего нужно определить отдаленную, *конечную* цель - какой результат вы, как планировщик проекта, ожидаете получить, когда экспертная система решит проблему. Недостаточно просто сказать, что "система должна решать производственные проблемы", так как сама по себе эта фраза не определяет, каким образом система будет приносить пользу предприятию. "Минимизация материальных затрат при увеличении выпуска продукции" - более точное выражение конечной цели. Цель должна выражать действие или событие, отображающее воздействие экспертной системы на общий ход событий. На этом уровне описания степень неопределенности может быть достаточно большой. В сценарии с пришельцами, описанном в прологе, можно поставить такую цель: "выбраться подобру-поздорову". В более практических применениях конечные цели могут быть выражены как "минимизировать налогообложение" или "увеличить выживание пациентов с недиагностированной волчанкой". Определенность возрастает при переходе на следующий уровень.

Определение промежуточных целей

Для каждой конечной цели система может иметь ряд *промежуточных* целей - действий или событий, которые произойдут после того, как будут решены частные проблемы. Бывают цели, достижение которых приводит к достижению конечных целей. В нашей экспертной системе освобождения одной из промежуточных целей было определение местонахождения двери лифта. В программе планирования налога вы можете попытаться минимизировать представляемый в декларации доход; в медицинской программе - диагностировать волчанку по клиническим проявлениям и истории болезни.

На этом уровне цели тоже выражаются как действия или события, но их неопределенность намеренно сокращается. Когда определены промежуточные цели, проблема распадается на подзадачи. Каждая цель представляет предполагаемый результат, предлагаемый программой при решении частной проблемы. Они должны устанавливаться в начале, так чтобы проектировщик имел ясное представление о задачах, которые должна решать система, и о подходе к их решению. Система может прийти к разнообразным выводам и рекомендовать различные стратегии поведения, и до тех пор, пока разработчик и пользователь не придут к соглашению, какие события должны быть выявлены в результате этих действий, не будет никаких гарантий в том, что получено нужное решение. Так как задача разработчиков состоит в том, чтобы наметить приемлемый путь к достижению этих целей-событий, они должны иметь конкретные представления об ожидаемых, ясно просматриваемых человеко-машинных характеристиках.

Кроме установления желаемых результатов решения проблемы, промежуточная цель выполняет и другую важную функцию. Вследствие своей специфической природы, она образует базис для расчленения общей проблемы на подпроблемы, которые могут потребовать для своего решения отдельных или изолированных систем знаний.

Определение вспомогательных целей

С самого начала планировщики должны иметь ясное представление о *вспомогательных* целях, совместное достижение которых потребуется для получения желаемых результатов. Вновь обратимся к сценарию освобождения из корабля пришельцев; для того чтобы "определить местонахождение шахты лифта", сначала нужно "определить свое местоположение". Вспомогательными целями здесь будут "определение этажа" и "определение направления север - юг". В нашем примере планирования налогов такие цели могут включать "максимизацию скидки за текущий год" и "максимизацию выплат с отсроченным налогом". Очевидно, что эта стадия планирования требует участия специалиста в предметной области на самом раннем этапе разработки.

Каждая вспомогательная цель предполагает содержание в базе знаний ряда связанных задач. Например, для определения направления север - юг пленник космического корабля инопланетян выглянет в иллюминатор, чтобы увидеть, откуда падает солнечный свет; одним из подходящих способов максимизации годовой скидки является применение метода УВС.

Определение проблем

После того как инженеры по знаниям определили цели экспертной системы, становятся достаточно очевидными типы проблем, которые предстоит решать, и способ подхода системы к их решению.

Определив проблемы, которые предстоит решать системе, группа разработчиков должна выяснить, возможно ли получение решений с помощью компьютера. Использует ли экспертиза, требующаяся для получения решения, выражения типа триплетов "объект - атрибут - значение"? Имеются ли достоверные связи между объектами? Является проблема по своей природе четкой или недоопределенной? Мы можем предположить, что проблема "Болен ли пациент волчанкой?" решается с помощью компьютера, при использовании конкретных анализов и перечня статистических корреляций. Но проблема "Правильно ли осуществляется руководство?" более неопределенна.

Кроме того, планировщик должен оценить, соответствуют ли проблемы конечным целям проекта. Ответ на вопрос "Кто является управляющим торговли в западном регионе?" не имеет практической важности. Более полезно было бы определить "Кто из управляющих при реализации продукции получает наивысший процент прибыли?".

Извлечение знаний

После определения целей и задач системы, группа разработчиков сталкивается с проблемой эффективного извлечения экспертных знаний. Наиболее очевидным методом является простой опрос. Интервьюирование эксперта или получение знаний из справочников - это прямой, но утомительный путь. Использование аналогий или моделей может значительно сократить эту работу: соберите файл экспертных решений, связанных с проблемами в рассматриваемой предметной области, затем проанализируйте главные правила, на которых основываются эти экспертные решения. Третий источник информации - прямое наблюдение или экспериментирование. Никому не нужен эксперт, чтобы установить, что "кровь красная", хотя эксперт может понадобиться для определения правила: "Если кровь темно-красная, то она насыщена кислородом".

Передача знаний - процесс итеративный и эволюционный. Вначале разработчик системы спрашивает эксперта о наиболее общих концепциях предметной области. По мере развития проекта общие контуры уточняются, обрастают связями и деталями. Для наглядности рассмотрим, как мог бы инженер по знаниям проводить опрос специалиста по освобождению пленника при создании экспертной системы, описанной в прологе:

ИНТЕРВЬЮЕР:	Если бы вы хотели выбраться из корабля, как бы вы приступили к разработке плана освобождения?
ЭКСПЕРТ:	Прежде всего я бы захотел установить свое местонахождение в корабле. Затем я бы хотел определить положение выхода относительно этого места.
ИНТЕРВЬЮЕР:	Как бы вы определили свое местонахождение?

ЭКСПЕРТ: Я бы узнал, на каком этаже и в какой части корабля - в северной или южной я нахожусь.

ИНТЕРВЬЮЕР: Как бы вы определили, в северной или южной части корабля вы находитесь?

ЭКСПЕРТ: Я бы попытался увидеть, какой солнечный свет попадает в иллюминатор, прямой (направление на юг) или отраженный (направление на север).

Классический метод извлечения экспертных знаний - пройти через все стадии процесса экспертизы, наблюдая, как в том или ином случае поступает эксперт. Допустим, что вы создаете программу для выработки рекомендаций относительно снижения налогов. Сначала вам следует изучить ситуации с различными демографическими и финансовыми характеристиками, представляющие типичные случаи. Затем следует попросить эксперта рекомендовать стратегию поведения в каждом случае, поясняя свои рекомендации. Предположим, эксперт рекомендует, во-первых, ограниченное участие в проекте нефтяного бурения, во-вторых, приобретение акций и, в-третьих, уход в отставку. Как инженеру по знаниям вам нужно понаблюдать, каким образом эксперт оценивает проблему и приходит к заключению, далее представить этот процесс в виде объектов, атрибутов, значений и правил - "схематичного знания" предметной области эксперта.

Пример

Для того чтобы продемонстрировать процесс преобразования экспертных знаний в доступные для машины факты и правила, рассмотрим простой пример. Допустим, что вы - древнегреческий философ, прогуливающийся вдоль берега Средиземного моря. Вдруг неожиданное искривление времени забрасывает вас в Манхэттен 20 века. Естественно, как знаменитый мудрец древности вы мгновенно становитесь знаменитостью и ваше имя - во всех списках приглашенных на модные сборища, театральные вечеринки и политические шоу Нью-Йорка. К несчастью, вы не имеете ни малейшего понятия, надо ли надевать галстук, и если да, то какого цвета.

Эта небольшая экспертная система придет вам на помощь. Выбрав консультанта по вопросам моды, вы начинаете извлечение экспертных знаний. Вы обнаруживаете, что галстук требуется, когда вы одеваете официальный костюм или деловой костюм в рабочий день. Если вы собираетесь поехать в деловом костюме на уикэнд, ваш консультант сообщает, что вы, если хотите, можете надеть галстук, но это необязательно. Далее эксперт объясняет, что если вы носите на работе спортивный пиджак, то чаще всего вам следует надевать галстук. Но если вы одеваете спортивный пиджак от случая к случаю, вряд ли тогда нужен галстук. К официальному костюму ваш консультант рекомендует черную бабочку или белый галстук. К голубому или серому деловому костюму больше подойдет красный галстук в полоску или иногда однотонный красный. Если костюм зеленого или коричневого цвета, ваш консультант по вопросам моды предлагает рыжий полосатый галстук, либо коричневый полосатый или, в крайнем случае, зеленый узорчатый галстук. Последующие рекомендации касаются типов и цветов галстуков для черных костюмов и серых, голубых, коричневых или зеленых пиджаков.

Для ввода информации в нашу экспертную систему, нужно создать текстовый файл с именем *rules*, содержащий базу знаний в форме вопросов, разрешенных значений и правил. Ниже показана база знаний для консультаций по вопросам моды.

```
правило1:    если
             костюм=официальный
        то
             надеть_галстук=да.
правило2:    если
             костюм=деловой и
             время=рабочий_день
        то
             надеть_галстук=да,кд=50.
правило3:    если
             костюм=деловой и
             время=уикэнд
        то
             надеть_галстук=да,кд=50.
```

```
правило4:    если
             костюм=спортивный_пиджак и
             занятие=работа
        то
             надеть_галстук=да,кд=75.
правило5:    если
             костюм=никакого
        то
             надеть_галстук=нет.
правило6:    если
             костюм=спортивный_пиджак и
             занятие=случайное
        то
             надеть_галстук=да,кд=10.
правило7:    если
             костюм=спортивный_пиджак и
             занятие=работа и
             время=вечер
        то
             надеть_галстук=да,кд=75.
правило8: если
             костюм=официальный и
             надеть_галстук=да
        то
             галстук=черная_бабочка,кд=50 и
             галстук=белый,кд=50.
правило9: если
             костюм=деловой и
             цвет_пиджака=голубой и
             надеть_галстук=да
        то
             галстук=рыжий_в_полоску,кд=75 и
             галстук=красный_однотонный,кд=35.
правило9а:   если
             костюм=деловой и
             цвет_пиджака=серый и
             надеть_галстук=да
        то
             галстук=рыжий_в_полоску,кд=75 и
             галстук=красный_однотонный,кд=35.
```

правило10: если
костюм=деловой и
цвет_пиджака=серый и
надеть_галстук=да
то
галстук=рыжий_в_полоску,кд=70 и
галстук=коричневый_в_полоску,кд=50 и
галстук=зеленый_узорчатый,кд=30.

правило10а: если
костюм=деловой и
цвет_пиджака=коричневый и
надеть_галстук=да
то
галстук=рыжий_в_полоску,кд=70 и
галстук=коричневый_в_полоску,кд=50 и
галстук=зеленый_узорчатый,кд=30.

правило11: если
костюм=деловой и
цвет_пиджака=черный и
надеть_галстук=да
то
галстук=красный_в_полоску,кд=60 и
желтый_в_горошек,кд=40.

правило12: если
костюм=спортивный_пиджак и
цвет_пиджака=коричневый и
надеть_галстук=да
то
галстук=рыжий_однотонный,кд=60 и
галстук=рыжий_в_полоску,кд=50 и
галстук=коричневый_с_рисунком,кд=40.

правило13: если
костюм=спортивный_пиджак и
цвет_пиджака=зеленый и
надеть_галстук=да
то
галстук=рыжий_однотонный,кд=60 и
галстук=рыжий_в_полоску,кд=50 и
галстук=коричневый_с_рисунком,кд=40.

```
разрешзн(костюм)=официальный,деловой,спортивный_пид-
                                    жак,никакого
разрешзн(время)=рабочий_день,уикэнд,вечер
разрешзн(занятие)=случайное,работа
разрешзн(цвет_пиджака)=голубой,серый,коричневый,зеленый,
                                    черный
вопрос(костюм)=Какой костюм вы оденете?
вопрос(занятие)=Чем вы заняты?
вопрос(цвет_пиджака)=Какого цвета пиджак вы оденете
                                    сегодня?
```

В этом файле правил имеются предложения четырех типов. Первый - это правила, которые выражают эмпирические связи между объектами и значениями в базе знаний. Каждое правило имеет такой синтаксис:

```
правилоN:    если
                ПРЕДПОСЫЛКА
          то
                ЗАКЛЮЧЕНИЕ.
```

Это предложение начинается с номера правила N, за которым следует двоеточие. Предпосылки и заключения составляют пары ОБЪЕКТ=ЗНАЧЕНИЕ. Заключение может, кроме того, иметь связанный с ним коэффициент доверия в виде ОБЪЕКТ=ЗНАЧЕНИЕ,кд=N.

Второй тип перечисляет разрешенные значения для имени объекта:

```
разрешзн(ОБЪЕКТ)=знач1,знач2,.....,значN
```

Это предложение начинается с ключевого слова "разрешзн", за которым следует заключенное в скобки имя объекта. В правой части равенства перечисляются имена разрешенных значений, разделенные запятыми.

Третий тип объявляет объект многозначным:
```
многозначный(ОБЪЕКТ)
```

Вопросы имеют следующий формат:
```
вопрос(ОБЪЕКТ)=ТЕКСТ
```

Это предложение начинается с ключевого слова "вопрос", за которым следует заключенное в скобки имя объекта. В правой части равенства записывается текст вопроса, который должен быть задан. Если текст содержит в конце вопросительный знак, не забудьте его поставить.

Обратите внимание, что все правила в файле *rules* оканчиваются точкой. Точка нужна для отделения одного правила от другого; без нее файлы представляются программе единым, довольно бессмысленным массивом символов.

Убедитесь, что вы назвали этот текстовый файл *rules* и поместите его на рабочий диск. Когда вы запустите скомпилированную программу Паскаля *expert*, она считает содержимое текстового файла *rules* и выведет на экран меню консультации.

И вы никогда больше не будете испытывать затруднений, раздумывая, какого цвета галстук вам надеть.

ГЛАВА 14

Взаимодействие с существующими приложениями

Сами по себе экспертные системы обладают ограниченными возможностями. Но когда мы рассматриваем объединение экспертных систем с существующими компьютерными приложениями, в голову приходят некоторые идеи очень интересных применений. Трудно назвать какие-либо применения персональных компьютеров, возможности которых нельзя бы было расширить с помощью дополнительной экспертизы; электронные таблицы, редакторы текстов, базы данных - вот минимальный перечень программ, которые можно сделать значительно более мощными, придав им возможность имитировать решение проблем человеком. Кроме того, гораздо проще включить существующую информацию в экспертную систему из базы данных, чем создавать новые многочисленные данные.

В этой главе мы приведем пример такого приложения - экспертная система взаимодействующая с базой данных. Рассмотрим систему оценки состояния здоровья, разработанную нами в предыдущих главах. Эта экспертная система выдаст свой диагноз после получения ответов пользователя на несколько вопросов. Некоторые из этих вопросов касаются привычек проверяемого, другие - его образа жизни. Еще одна группа вопросов попадает в категорию истории болезни и статистических сведений - информации, которая может содержаться в компьютерной базе данных.

В другом примере экспертная система может быть объединена с программой электронной таблицы для создания программы "разумного критика", советчика по финансовым вопросам, способного вырабатывать рекомендации на основе накопленной информации. Допустим, вы используете Lotus 1-2-3 или SuperCalc для представления суммы скидки для приобретения имущества фирме. Пользуясь фактами и правилами

для планирования налогов, ваша "экспертная электронная таблица" может выбрать наилучший метод для каждого вида имущества вместо простого вычисления вычитаемой суммы.

Оба эти примера требуют наличия у системы важного свойства: возможности экспертной системы получать информацию из файла данных независимого приложения. К сожалению, каждое приложение применяет свой собственный метод хранения данных. Экспертная система должна быть специально доработана для чтения внешних файлов данных.

Использование экспертной системы с файлами базы данных

Рассмотрим метод извлечения информации из файла базы данных, созданного с помощью популярной системы управления базами данных фирмы Ashton-Tate, dBASE III, чтобы проиллюстрировать вступление. В приведенном примере файл данных состоит из полей для пяти имен объектов, использованных в нашей гипотетической программе определения состояния здоровья. Эти поля содержат данные о фамилии человека, возрасте, поле, весе, происхождении и расе. Мы создадим для нашей экспертной системы специальный модуль, извлекающий записи, хранящиеся в этих полях в виде текста в коде ASCII.

Вот как работает такая система.

Допустим, что нужные записи хранятся в файле SUBJECTS, созданном в dBASE II или dBASE III, со следующими полями: NAME, AGE, GENDER, WEIGHT, ORIGIN, RACE. Если вы хотите сами испробовать работу этой подсистемы, создайте файл dBASE с перечисленными полями записи. Для подготовки этого файла к объединению с экспертной системой выполните следующие команды:

1. use (имя файла)
2. copy to SUBJECTS delimited with '.

Каждая строка этого файла будет содержать текст одной записи. Поля записи заключены в апострофы и разделены запятыми. Например, файл SUBJECTS может содержать следующие типичные записи:

'Салли Шепард','40','ж','110','с_амер','европеоид'
'Роберт Морган','25','м','150','средизем','негроид'
'Вики Шварц','65','ж','90','с_амер','европеоид'

Если в вашем распоряжении нет dBASE III (или dBASE II), просто создайте файл в этом формате, используя текстовый редактор.

Модуль, приведенный в качестве примера в настоящей главе, позволяет пользователю до начала консультации с экспертной системой ввести имя человека. Система в ответ вызывает файл с именем SUBJECTS и ищет в нем статистические данные об указанном лице. Затем программа поле за полем вводит соответствующие данные в базу знаний. Когда все значения считаны, начинается обычная консультация. Однако система не будет запрашивать у оператора значения, которые уже выбраны из файла данных.

Управление динамическими данными

Прежде чем создавать этот дополнительный модуль, давайте рассмотрим более сложный пример. Допустим, вы являетесь консультантом в бюро путешествий, создающим экспертную систему для рекомендаций наилучшего способа воздушного сообщения между двумя любыми городами, учитывающую надежность, стоимость, питание и число остановок. Записи в файле такого типа, по-видимому, будут постоянно меняться, отражая текущие изменения расписания полетов, стоимости пассажирских перевозок и наличия мест в каждом классе. В то время как запись о медицинских показаниях может содержать только одно значение для пола или расы, запись о воздушных перевозках будет подвержена постоянным изменениям при продаже билетов, росте или снижении цен и отмене или переносе рейсов. При непосредственном взаимодействии экспертной системы с базой данных, ноу-хау программы становится таким же динамичным, как и у эксперта.

Допустим, файл воздушных сообщений содержит следующие поля: САМОЛЕТ, ПАРА_ГОРОДОВ, П_МЕСТ, С_МЕСТ, П_ЦЕНА, ПИТАНИЕ, ОСТАНОВКИ. Буква П в имени поля обозначает "первый класс", а С - "салон". Записанный в формате dBASE, файл может выглядеть примерно так:

'AA','СФОМАЙ','4П','4С','475','305','З','1'
'UA','БУФАТЛ','4П','0С','385,','295','О','0'
'TW','ТУЛСИЭ','4П','4С','320','245','У','2'

Записи должны быть введены в файл с использованием имен значений, понятных для экспертной системы; или, наоборот, имена, присвоенные значениям, должны соответствовать элементам файла данных. Например, если экспертная система применяет коды аэропортов, такие, как СФО для Сан-Франциско или МАЙ для Майами, то в поле ПАРА_ГОРОДОВ следует ввести СФОМАЙ, а не Сан-Франциско - Майами. Аналогично, если в базе данных слово "завтрак" обозначается символом З, одним из разрешенных значений поля ПИТАНИЕ будет З, а не ЗАВТРАК.

В рассмотренном примере кодовые обозначения использованы для авиакомпаний (AA для Американской, UA для Объединенной, TW для TWA) и для аэропортов. Первое поле каждой записи отведено для авиакомпании, затем идут пары городов (коды для аэропортов "отправления" и "прибытия"), наличие мест в первом классе, наличие мест в общем салоне, питание и число остановок. Когда экспертная система считывает поле для Объединенной компании, она должна опознать 4П как разрешенное значение объекта П_МЕСТ, а 0С как разрешенное значение С_МЕСТ. Вводя в начале консультации тип самолета, вы должны вместо полного наименования набирать на клавиатуре коды: AA, UA или TW.

Программные модули

Этот модуль ищет текстовый файл с именем SUBJECTS, содержащий записи, которые будут использоваться в экспертной системе. Наш пример предназначен для применения в системе оценки состояния здоровья, рассмотренной в предыдущих главах. Процедура IMPORT сначала запрашивает у оператора имя пациента. Программа считывает текст, введенный с клавиатуры, обращается к файлу SUBJECTS и находит запись, содержащую заданное имя.

Затем процедура начинает читать поля найденной записи файла. Предложение CASE 2 добавляет второе поле (возраст человека) в базу знаний. Предложение CASE 3 добавляет третье поле (пол). Программа продолжает работать таким образом до тех пор, пока не будет достигнута метка

конца файла. Если запись, содержащая указанное имя, не найдена, программа выводит на экран сообщение "имя не найдено".

```
PROCEDURE import;

VAR
dbase:Text;
d_field,
name,
word:word_string;
line:line_string;
done:boolean;
counter:integer;

BEGIN
writeln;
writeln('Введите фамилию');
readln(name);
assign(dbase,'subjects');
reset(dbase);
word:='';
writeln;

WHILE ((NOT Eof(dbase))AND(word<>name)) DO
        BEGIN
        readln(dbase,line);
        done:=find_word(line,1,word);
        word:=(copy(word,2,length(word)-2)
        END;

IF (word=name) THEN
        BEGIN
        done:=FALSE;
        counter:=1;
        WHILE (done=FALSE) DO
                BEGIN
                done:=find_word(line,counter,word);
                word:=copy(word,2,length(word)-2);

                CASE counter OF
```

```
                2:
                BEGIN
                d_field:='возраст';
                add_object(d_field,word);
                add_cf(d_field,word,DEFINITE)
                END;

                3:
                BEGIN
                d_field:='пол';
                add_object(d_field,word);
                add_cf(d_field,word,DEFINITE);
                END;

                4:
                BEGIN
                d_field:='вес';
                add_object(d_field,word);
                add_cf(d_field,word,DEFINITE);
                END;

                5:
                BEGIN
                d_field:='происхождение';
                add_object(d_field,word);
                add_cf(d_field,word,DEFINITE);
                END;

                6:
                BEGIN
                d_field:='раса';
                add_object(d_field,word);
                add_cf(d_field,word,DEFINITE);
                END
                END;
                counter:=counter+1
        END
        END
ELSE writeln('фамилия не найдена.')

END;
```

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

ПОЛНЫЙ ЛИСТИНГ ОСНОВНОЙ ПРОГРАММЫ

Полная программа *expert* считывает всю базу знаний из текстового файла *rules*, находящегося на принимаемом по умолчанию запоминающем устройстве. Следовательно, различные возможности, которые предлагаются в меню предыдущих версий основной программы, рассмотренных в этой книге, становятся больше не нужны. Подготовьте базу знаний, следуя инструкциям гл. 13 "Использование экспертных систем".

Приведенный ниже листинг включает в себя все необходимые описания типов и глобальных переменных. Однако в нем не дублируются функции и процедуры, содержащиеся в предшествующих главах.

Как ввести программу

Программу на Паскале можно представить как состоящую из трех отдельных частей:

1. Определения типов и глобальных переменных.
2. Функции и процедуры.
3. Тело основной программы.

Сначала вам нужно ввести заголовок программы:

```
Programm Expert(Input,Output,rules,dbase);
```

Прежде чем вводить какие-либо модули, наберите определения типов и глобальных переменных. Это - части листинга, идущие под заголовками CONST, TYPE и VAR. Затем добавьте модули функций и процедур в том порядке, в каком они перечислены в этой книге. Будьте внимательны, старайтесь не пропустить какие-нибудь модули, так как вновь

вводимые подпрограммы часто обращались к модулям, введенным ранее.

В заключение наберите текст основной программы в том виде в каком она представлена в нашем приложении - начиная с заголовка BEGIN и кончая последним предложением END. Убедитесь, что последним введенным символом является точка и не перепутайте два соседних апострофа с кавычками.

```
Program Expert(Input,Output,rules,dbase);


CONST
        WORD_MAX=40; - max длина имён
        LINE_MAX=80; - max длина строки
        COLON=':';
        PERIOD='.';
        COMMA=',';
        SPACE=' ';
        EQUALS='=';
        DEFINITE=100;

TYPE
        word_string=string[WORD_MAX]; - тип имен для объектов и значений
        linc_string=string[LINE_MAX]; - тип строки
        value_ptr=^value; - указатель на значение
        legal_ptr=^legal_value;
        object_ptr=^object; - указатель на объект

        value=RECORD
                name:word_string; имя значения
                cert:integer;
                setby:word_string;
                ncxt:value_ptr - указатель на следующ. значение
                END;

        legal_value=RECORD
                name:word_string;
                ncxt:legal_ptr
                END;
```

```
object=RECORD
        name:word_string;
        question:linc_string;
        multivald:boolean;
        legal_list:legal_ptr;
        sought:boolean;
        valuc_list:valuc_ptr;
        next:objcct_ptr
        END;

prem_ptr=^prem;
con_ptr=^con;
rule_ptr=^rule;

prem=RECORD
        object:word_string;
        value:word_string;
        next:prem_ptr
        END;

con=RECORD
        object:word_string;
        value:word_string;
        cert:integer;
        next:con_ptr
        END;

rule=RECORD
        name:word_string;
        prem:prem_ptr;
        con:con_ptr;
        next:rule_ptr
        END;

VAR
        choice,
        max_choice,
        choice_lim:integer;
        last_try,
        top_fact:object_ptr;
        s_word,
        s_object,
```

```
    s_value:word_string;
    s_line:line_string;
    s_cf:integer;
    top_rule:rule_ptr;
    rules:Text;
    explain:boolean;
```

```
BEGIN
last_try:=NIL;
top_fact:=NIL;
choice:=0;
import;
read_file;
writeln;
writeln;
writeln('Хотите ли вы получить объяснение хода вывода?');
writeln('1 - да');
writeln('2 - нет');
readln(choice);
IF (choice=1) THEN explain:=TRUE
ELSE explain:=FALSE;
writeln('Какова ваша цель ?');
writeln('(Введите:(ОБЪЕКТ))');
readln(s_object);
pursue(s_object);
p_result(s_object)
END.
```

ПРИЛОЖЕНИЕ Б

ПЕРЕНОСИМОСТЬ

Изменения для работы программы на APPLE 1.1 или UCSD - Паскале

В UCSD - Паскале нет фупкции VAL, входящей в функцию GET_CF, преобразующую строку в целое число, поэтому она должна быть опущена. Это не позволит пользователю применять в выражениях коэффициента доверия числа. Вместо них следует использовать пять ключевых слов:

```
BEGIN
trim:=copy(f_line,st_right+3,length(f_line)-st_right-2);
val(trim,cf,result);
IF (result>0) THEN cf:=DEFINITE;
```

Замените подчеркнутые строки на следующую:

```
cf:=DEFINITE;
```

В процедуре READ_FILE команду ASSIGN следует опустить, а команду RESET - изменить.

```
BEGIN
writeln('Чтение файла, содержащего правила');
assign(rules,'rules');
reset(rules,'rules);
```

Удалите подчеркнутую строку.

В процедуре IMPORT команду ASSIGN следует опустить, а команду RESET - изменить.

```
BEGIN
writeln;
writeln('Введите фамилию');
readln(name);
assign(dbase,'subjects');
reset(dbase,'subjects);
```

Удалите подчеркнутую строку.

ПРИЛОЖЕНИЕ В

ПРИМЕРЫ БАЗ ЗНАНИЙ

Ниже в качестве примера приведены две базы знаний, которые можно использовать с основной программой. Первая выдаст рекомендации по конфигурации деловой компьютерной системы на основе информации об области применения, позволяя пользователю принять или отвергнуть предложения системы. Вторая предназначена для выработки советов на основе оценки состояния здоровья, описанной в предыдущих главах. Она определяет предполагаемую продолжительность жизни человека на основании полученной о нем информации. Советы, получаемые с помощью этих систем, служат только демонстрационным целям.

Несмотря на большое число правил в каждой системе, иногда может оказаться, что системе не хватает информации для выдачи рекомендаций.

Конфигурация компьютерной системы

Инструкции:

1. Напечатайте базу знаний и сохраните ее в файле с именем "rules".

2. Запустите программу экспертной системы и укажите цель "выполнено".

разрешзн(тип)=производство,распределение
разрешзн(тип)=получение,продажа,сервис,финансы
разрешзн(пр_кат)=индустрия,ширпотреб
разрешзн(инд_кат)=тяж_пром,лег_пром
разрешзн(ширп_кат)=прод_питания,одежда,электроприборы
разрешзн(ширп_кат)=мебель,прочее
разрешзн(пазн_кат)=оптовая_торг,транспорт
разрешзн(пазн_кат)=пассажиры,импорт_экспорт
разрешзн(опт_кат)=склад,без_склада
разрешзн(трансп_кат)=возд,жел_дор,автомоб,водн
разрешзн(пасс_кат)=самолет,поезд,такси,автобус,судно
разрешзн(пр_кат)=сельхоз,пол_ископаемые
разрешзн(сельх_кат)=скот,товар,спец
разрешзн(иск_кат)=нефть_газ,уголь,минералч,прочее
разрешзн(прод_кат)=общ,продукты,спец
разрешзн(серв_кат)=врач_дантист,бухгалтер,юрист
разрешзн(серв_кат)=рекл_агент,архитектор
разрешзн(фин_кат)=банки,недвижимость,
разрешзн(фин_кат)=инвестиции,консультации
разрешзн(годовой_оборот)=свыше_5млн,свыше_10млн
разрешзн(годовой_оборот)=свыше_20млн,другой

разрешзн(число_служ)=мен_100,100-500,бол_500
разрешзн(число_клиент)=мен_10,10-25,25-100
разрешзн(число_клиент)=100-500,бол_500
разрешзн(число_поставщиков)=мен_10,10-25,25-100
разрешзн(число_поставщиков)=100-500,бол_500
разрешзн(средн_оборот)=мен_10,10-100,100-1000
разрешзн(средн_оборот)=1000-10000,бол_10000
разрешзн(возраст_фирмы)=мен_1_г,1-3_г,3-10_л
разрешзн(возраст_фирмы)=10-50_л,бол_50_л
разрешзн(масштаб)=город,регион,страна
разрешзн(масштаб)=континент,мир
разрешзн(перв_ком)=да,нет
разрешзн(цель)=общая,специальная
разрешзн(сп_цель)=общие,расчеты,подготовка_текстов
разрешзн(сп_цель)=управл_торгов,разработка,планирование
разрешзн(сп_расч)=зап_расч,увольнения,зарплата
разрешзн(сп_расч)=общ_регистр,управл_инвест
разрешзн(сп_текст)=корреспонд,записи,распоряжения

```
разрешизн(сп_текст)=черновой,профессиональный
разрешизн(сп_разр)=продукция,предложения
разрешизн(сп_разр)=реклама,представление
разрешизн(макс_цена)=мсп_5000,5000-10000,бол_10000

вопрос(тип)=Какого типа бизнесом вы занимаетесь?
вопрос(пр_кат)=Какой тип продукции вы производите?
вопрос(инд_кат)=Какие индустриальные товары вы выпускаете?
вопрос(ширп_кат)=Какого типа ширпотреб вы выпускаете?
вопрос(назн_кат)=В какой области промышленности вы работаете?
вопрос(опт_кат)=Каков способ оптовой торговли?
вопрос(трансп_кат)=Какой используется транспорт?
вопрос(пасс_кат)=Какой вид пассажирского транспорта?
вопрос(пр_кат)=В каком виде производства вы заняты?
вопрос(сельх_кат)=Какой вид сельского хозяйства?
вопрос(иск_кат)=Какого типа полезные ископаемые?
вопрос(прод_кат)=Какие у вас магазины?
вопрос(серв_кат)=Каков род ваших занятий?
вопрос(фин_кат)=Каков род ваших занятий?

вопрос(средн_оборот)=Каков ваш средний оборот?
вопрос(число_служ)=Каково число ваших служащих?
вопрос(число_клиент)=Сколько у вас клиентов в год?
вопрос(число_поставщиков)=Сколько у вас поставщиков?
вопрос(средн_оборот)=Каков ваш средний оборот?
вопрос(возраст_фирмы)=Сколько времени вы ведете дело?
вопрос(масштаб)=Вашим рынком является:
вопрос(перв_ком)=Это ваш первый компьютер?
вопрос(цель)=Он предназначен для специальной цели?
вопрос(сп_цель)=Для какого применения он предназначен?
вопрос(сп_расч)=Для каких расчетов?
вопрос(сп_текст)=Для подготовки каких текстов?
вопрос(сп_разр)=Для каких разработок?
вопрос(макс_цена)=Сколько вы можете потратить?
вопрос(выполнено)=Конец консультации - НАЖМИТЕ ENTER

многозначный(выполнено)
```

```
правило0: если
        запуск=да
то
        выполнено=да.

вопрос(запуск)=Начало системы "ВЫБОР". НАЖМИТЕ ENTER

правило1: если
        тип=производство и
        пр_кат=индустрия и
        инд_кат=найдено
то
        выполнено=да.

правило2: если
        тип=производство и
        пр_кат=ширпотреб и
        ширп_кат=найдено
то
        выполнено=да.

правило3: если
        тип=распределение и
        назн_кат=оптовая_торг и
        опт_кат=найдено
то
        выполнено=да.

правило4: если
        тип=распределение и
        назн_кат=транспорт и
        трансп_кат=найдено
то
        выполнено=да.

правило5: если
        тип=распределение и
        назн_кат=пассажиры и
        пасс_кат=найдено
то
        выполнено=да.
```

```
правило6: если
        тип=получение и
        пр_кат=сельхоз и
        сельх_кат=найдено
то
        выполнено=да.

правило7: если
        тип=получение и
        пр_кат=пол_ископаемые и
        иск_кат=найдено
то
        выполнено=да.

правило8: если
        тип=продажа и
        прод_кат=найдено
то
        выполнено=да.

правило9: если
        тип=сервис и
        серв_кат=найдено
то
        выполнено=да.

правило10: если
        тип=финансы и
        фин_кат=найдено
то
        выполнено=да.

правило11: если
        число_служ=найдено
то
        выполнено=да.

правило12: если
        число_клиент=найдено
то
        выполнено=да.
```

```
правило13: если
        число_поставщиков=найдено
то
        выполнено=да.

правило14: если
        средн_оборот=найдено
то
        выполнено=да.

правило15: если
        возраст=фирмы=найдено
то
        выполнено=да.

правило16: если
        масштаб=найдено
то
        выполнено=да.

правило17: если
        перв_ком=найдено
то
        выполнено=да.

правило18: если
        сп_цель=расчеты и
        сп_расчет=найдено
то
        выполнено=да.

правило19: если
        сп_цель=подготовка_текстов и
        сп_текст=найдено
то
        выполнено=да.

правило20: если
        сп_цель=разработка и
        сп_разр=найдено
то
        выполнено=да.
```

```
правило21: если
        макс_цена =найдено
то
        выполнено =да.

правило22: если
        число_служ =100-500
то
        чис_сл =большое.

правило23: если
        число_служ =бол_500
то
        чис_сл =большое.

правило24: если
        число_служ =мен_100
то
        чис_сл =небольшое.

правило25: если
        макс_цена =5000-10000 и
        п1 =да
то
        цпу =xt.

вопрос(п1)=Я предлагаю компьютер XT, вас устраивает?
разрешзн(п1)=да,нет

правило26: если
        сп_цель=общие и
        макс_цена =бол_10000 и
        п1а =да
то
        цпу =at.

вопрос(п1а)=Я предлагаю компьютер AT, вас устраивает?
разрешзн(п1а)=да,нет
```

```
правило27: если
        п16=да
то
        цпу=рс.

вопрос(п16)=Я предлагаю компьютер РС, вас устраивает?
разрешзн(п16)=да,нет

правило28: если
        годовой_оборот=свыше_20млн и
        сп_цель=расчеты и
        п2=да
то
        осн_память=40м_тверд_диск.

правило29: если
        годовой_оборот=свыше_20млн и
        сп_цель=разработка и
        п2=да
то
        осн_память=40м_тверд_диск.

вопрос(п2)=Я предлагаю твердый диск на 40 м, вас устраивает?
разрешзн(п2)=да,нет

правило30: если
        годовой_оборот=свыше_10млн и
        сп_цель=расчеты и
        п3=да
то
        осн_память=20м_тверд_диск.

правило31: если
        число_клиент=бол_500 и
        сп_цель=управл_торг и
        п3=да
то
        осн_память=20м_тверд_диск.

вопрос(п3)=Я предлагаю твердый диск на 20 м, вас устраивает?
разрешзн(п3)=да,нет
```

```
правило32: если
        тип=финансы и
        п4=да
то
        память=512K.

правило33: если
        сп_цель=планирование и
        п4=да
то
        память=512K.

правило34: если
        сп_цель=разработка и
        п4=да
то
        память=512K.

вопрос(п4)=Я предлагаю машину 512K, согласны?
разрешизн(п4)=да,нет

правило35: если
        сп_цель=разработка и
        макс_цена=бол_10000 и
        п5=да и
        п6=да
то
        плоттер=большой_цветной и
        монитор=выс_разреш_цветной.

вопрос(п5)=Я предлагаю большой цветной плоттер, согласны?
разрешизн(п5)=да,нет

правило36: если
        сп_цель=разработка и
        макс_цена=бол_10000 и
        п6=да и
        п7=да
то
        плоттер=малый_цветной и
        монитор=выс_разреш_цветной.
```

вопрос(п7)=Я предлагаю малый цветной плоттер, согласны?
разрешзн(п7)=да,нет

правило37: если
тип=сервис и
серв_кат=рекл_агент и
макс_цена=бол_10000 и
п6=да
то
монитор=выс_разреш_цветной.

правило38: если
тип=сервис и
серв_кат=архитектор и
макс_цена=бол_10000 и
п6=да
то
монитор=выс_разреш_цветной.

вопрос(п6)=Я предлагаю цветной монитор выс.раз, согласны?
разрешзн(п6)=да,нет

правило39: если
годовой_оборот=свыше_5млн и
сп_цель=расчеты и
п8=да
то
осн_память=10м_тверд_диск.

правило40: если
макс_цена=5000-10000 и
п8=да
то
осн_память=10м_тверд_диск.

вопрос(п8)=Я предлагаю твердый диск на 10 м, вас устраивает?
разрешзн(п8)=да,нет

```
правило41: если
        сп_цель=расчеты и
        п9=да
то
        по=общее.

вопрос(п9)=Я предлагаю общее прогр. обеспечение, согласны?
разрешзн(п9)=да,нет


правило42: если
        п10=да
то
        память=256К.


правило43: если
        цпу=найдено
то
        выполнено=да.


правило44: если
        память=найдено
то
        выполнено=да.


правило45: если
        осп_память=найдено
то
        выполнено=да.

правило46: если
        архивп_пам=найдено
то
        выполнено=да.

правило47: если
        монитор=найдено
то
        выполнено=да.
```

```
правило48: если
        принтер=найдено
то
        выполнено=да.

правило49: если
        по=найдено
то
        выполнено=да.

правило50: если
        wp=найдено
то
        выполнено=да.

правило51: если
        db=найдено
то
        выполнено=да.

правило52: если
        сист_план=найдено
то
        выполнено=да.

правило53: если
        граф_сист=найдено
то
        выполнено=да.

вопрос(п10)=Я предлагаю машину с 256К, согласны?
разрешзн(п10)=да,нет

правило54: если
        п11=да
то
        осн_память=флопи-диск.

вопрос(п11)=Я предлагаю флопи-диск,.согласны?
разрешзн(п11)=да,нет
```

```
правило55: если
        осн_память=40м_тверд_диск и
        п12=да
то
        архивн_память=магн_лента.

вопрос(п12)=Я предлагаю архив на магнитной ленте, согласны?
разрешизн(п12)=да,нет

правило56: если
        осн_память=40м_тверд_диск и
        п13=да
то
        архивн_память=флоппи-диск.

вопрос(п12)=Я предлагаю архив на флоппи-диске, согласны?
разрешизн(п12)=да,нет

правило57: если
        п14=да
то
        монитор=моно_выс_разр.

вопрос(п14)=Я предлагаю монитор моно с выс. разр., согласны?
разрешизн(п14)=да,нет

правило58: если
        сп_цель=подготовка_текстов и
        сп_текст=распоряжения и
        п15=да
то
        принтер=буквенный.

правило59: если
        сп_цель=подготовка_текстов и
        сп_текст=корреспонд и
        п15=да
то
        принтер=буквенный.

вопрос(п15)=Я предлагаю буквенный принтер, согласны?
разрешизн(п15)=да,нет
```

```
правило60: если
        п16=да
то
        принтер=матричный.

вопрос(п16)=Я предлагаю матричный принтер, согласны?
разрешзн(п16)=да,нет


правило61: если
        сп_цель=расчеты и
        п17=да
то
        по=спец.


правило62: если
        сп_цель=общие и
        п17=да
то
        по=спец.

вопрос(п17)=Я предлагаю специальное ПО, согласны?
разрешзн(п17)=да,нет


правило63: если
        сп_цель=общие и
        п18=да
то
        wp_сист=мощная.


правило64: если
        сп_цель=подготовка_текстов и
        п18=да
то
        wp_сист=мощная.

вопрос(п18)=Я предлагаю редактор текстов, согласны?
разрешзн(п18)=да,нет
```

```
правило65: если
        сп_цель=управл_торг и
        п19=да
то
        db_сист=мощная.

правило66: если
        сп_цель=управл_торг и
        п19=да
то
        db_сист=мощная.

вопрос(п19)=Я предлагаю мощную СУБД, согласны?
разрешзн(п19)=да,нет

правило67: если
        сп_цель=общие и
        п20=да
то
        db_сист=простая.

вопрос(п20)=Я предлагаю простую СУБД, согласны?
разрешзн(п20)=да,нет

правило68: если
        db_сист=файловая и
        п21=да
то
        wp_сист=простая и
        граф_сист=простая.

вопрос(п21)=Я предлагаю простой графич. пакет, согласны?
разрешзн(п21)=да,нет

правило69: если
        плоттер=найден и
        п22=да
то
        граф_сист=мощная.

вопрос(п22)=Я предлагаю мощный графич. пакет, согласны?
разрешзн(п22)=да,нет
```

```
правило70: если
        сп_цель=планирование и
        п23=да
то
        сист_план=электронная_таблица.

правило71: если
        сп_цель=общие и
        п23=да
то
        сист_план=электронная_таблица.

вопрос(п23)=Я предлагаю электронную таблицу, согласны?
разрешен(п23)=да,нет

вопрос(цпу)=тип компьютера неизвестен, НАЖМИТЕ ENTER
вопрос(память)=размер памяти неизвестен, НАЖМИТЕ ENTER
вопрос(осп_память)=осн. память неизвестна, НАЖМИТЕ ENTER
вопрос(архив)=архивн. память неизвестна, НАЖМИТЕ ENTER
вопрос(монитор)=монитор неизвестен, НАЖМИТЕ ENTER
вопрос(принтер)=принтер неизвестен, НАЖМИТЕ ENTER
вопрос(по)=система ПО неизвестна, НАЖМИТЕ ENTER
вопрос(wp_сист)=редактор текста неизвестен, НАЖМИТЕ ENTER
вопрос(db_сист)=СУБД неизвестна, НАЖМИТЕ ENTER
вопрос(сист_план)=элект. табл. неизвестна, НАЖМИТЕ ENTER
вопрос(граф_сист)=граф. система неизвестна, НАЖМИТЕ ENTER
вопрос(плоттер)=плоттер неизвестен, НАЖМИТЕ ENTER
```

Пример работы базы знаний конфигуратора компьютерной системы

```
Какова ваша цель?
        (Введите (ОБЪЕКТ))
выполнено
Начало системы "ВЫБОР". НАЖМИТЕ ENTER
```

Какого типа бизнесом вы занимаетесь?
1. производство
2. распределение
3. получение
4. продажа
5. сервис
6. финансы
1

Какой тип продукции вы производите?
1. индустрия
2. ширпотреб
2

Какого типа ширпотреб вы выпускаете?
1. прод_питания
2. одежда
3. электроприборы
4. мебель
5. прочее
2

Каково число ваших служащих?
1. мен_100
2. 100-500
3. бол_500
1

Сколько у вас клиентов в год?
1. мен_10
2. 10-25
3. 25-100
4. 100-500
5. бол_500
2

Сколько у вас поставщиков?
1. мен_10
2. 10-25
3. 25-100

4. 100-500
5. бол_500
1

Каков ваш средний оборот?
1. мен_10
2. 10-100
3. 100-1000
4. 1000-10000
5. бол_10000
2

Сколько времени вы ведете дело?
1. мен_1_г
2. 1-3_г
3. 3-10_л
4. 10-50_л
5. бол_50_л
1

Вашим рынком является:
1. город
2. регион
3. страна
4. континент
5. мир
3

Это ваш первый компьютер?
1. да
2. нет
2

Для какого применения он предназначен?
1. общее
2. расчеты
3. подготовка_текстов
4. управл_торгов
5. разработка
6. планирование
7.
1

Сколько вы можете потратить?
1. мен_5000
2. 5000-10000
3. бол_10000
2

Я предлагаю компьютер XT, вас устраивает?
1. да
2. нет
1

Я предлагаю машину с 256K, согласны?
1. да
2. нет
1

Каков ваш годовой оборот?
1. свыше_5млн
2. свыше_10млн
3. свыше_20млн
4. другой
2

Я предлагаю твердый диск на 10 м, вас устраивает?
1. да
2. нет
1

архивн. память неизвестна, НАЖМИТЕ ENTER

Я предлагаю монитор моно с выс. разр., согласны?
1. да
2. нет
1

Я предлагаю матричный принтер, согласны?
1. да
2. нет
1

Я предлагаю специальное ПО, согласны?
1. да
2. нет
1

Я предлагаю редактор текстов, согласны?
1. да
2. нет
1

Я предлагаю простую СУБД, согласны?
1. да
2. нет
1

Я предлагаю электронную таблицу, согласны?
1. да
2. нет
1

плоттер неизвестен, НАЖМИТЕ ENTER

граф. система неизвестна, НАЖМИТЕ ENTER

Конец консультации - НАЖМИТЕ ENTER

РЕЗУЛЬТАТ КОНСУЛЬТАЦИИ:
выполнено=да,кд=100

КОНЕЦ КОНСУЛЬТАЦИИ

Оценка состояния здоровья

Инструкции:

1. Напечатайте базу знаний и сохраните ее в файле с именем "rules".

2. Запустите программу экспертной системы и укажите цель "продолжительность".

База знаний оценки состояния здоровья

```
разрешен(старт)=да,выход
разрешен(вес)=55_или_меньше,55-85,85_или_больше
разрешен(сложение)=мелкое,крупное
разрешен(жиры)=норма,излишек
разрешен(холестерин)=норма,излишек
разрешен(соль)=норма.излишек
разрешен(раса)=негроидная,европеоидная,монголоидная
разрешен(происхождение)=американское,средиземноморское
разрешен(потребление_алкоголя)=не_употребляет,умеренное
разрешен(потребление_алкоголя)=чрезмерное
разрешен(пол)=м,ж
разрешен(возраст)=25_или_меньше,25_55,55_или_больше
разрешен(характер)=агрессивный,мягкий
разрешен(курение)=да,нет

вопрос(старт)=Вы готовы начать?
вопрос(продолжительность)=ИЗВИНИТЕ, ОТВЕТ НЕИЗВЕС-
                    ТЕН  -  НАЖМИТЕ ENTER
вопрос(вес)=Каков вес испытуемого?
вопрос(сложение)=Телосложение:
вопрос(жиры)=Потребление ненасыщенных жиров:
вопрос(холестерин)=Потребление животных жиров:
вопрос(соль)=Потребление соли:
вопрос(раса)=Какой расы испытуемый?
вопрос(происхождение)=Каково происхождение испытуемого?
вопрос(потребление_алкоголя)=Потребление алкоголя:
вопрос(пол)=Какого пола испытуемый?
вопрос(возраст)=Сколько испытуемому лет?
вопрос(курение)=Испытуемый курит?
вопрос(характер)=Характер испытуемого:

правило1: если
        относительный_вес=нормальный
то
        зн=да.

правило2: если
        относительный_вес=ниже_нормы
то
        зн=да.
```

правило3: если
коронарный_риск =ниже_среднего
то
сердзаб =низкий.

правило4: если
коронарный_риск =средний
то
сердзаб =низкий.

правило5: если
старт =да
то
сердзаб =иной.

правило6: если
возраст =25_или_меньше и
пол =ж
то
основная_продолжительность =72.

правило7: если
возраст =25_или_меньше и
пол =м
то
основная_продолжительность =67.

правило8: если
возраст =25-55 и
пол =ж
то
основная_продолжительность =67.

правило9: если
возраст =25-55 и
пол =м
то
основная_продолжительность =62.

правило10: если
возраст=55_или_больше и
пол=ж
то
основная_продолжительность=64.

правило11: если
возраст=55_или_больше и
пол=м
то
основная_продолжительность=60.

правило12: если
вес=55_или_меньше и
сложение=мелкое и
пол=ж
то
относительный_вес=нормальный.

правило13: если
вес=85_или_больше и
сложение=мелкое и
пол=ж
то
относительный_вес=излишний.

правило14: если
вес=55-85 и
сложение=мелкое и
пол=ж
то
относительный_вес=излишний.

правило15: если
вес=55_или_меньше и
сложение=крупное и
то
относительный_вес=недостаточный.

правило16: если
вес=55_или_меньше и
сложение=крупное и
пол=м
то
относительный_вес=недостаточный.

правило17: если
вес=55-85 и
сложение=мелкое и
пол=ж
то
относительный_вес=излишний.

правило18: если
вес=55-85 и
сложение=крупное и
пол=ж
то
относительный_вес=нормальный.

правило19: если
вес=55-85 и
сложение=мелкое и
пол=м
то
относительный_вес=нормальный.

правило20: если
вес=55-85 и
сложение=крупное и
пол=м
то
относительный_вес=недостаточный.

правило21: если
вес=85_или_больше и
пол=ж
то
относительный_вес=излишний.

правило22: если
вес=85_или_больше и
пол=м и
сложение=мелкое
то
относительный_вес=излишний.

правило23: если
вес=85_или_больше и
пол=м и
сложение=крупное
то
относительный_вес=нормальный.

правило24: если
холестерин=низкий и
жиры=излишек
то
коронарный_риск=ниже_среднего.

правило25: если
холестерин=низкий и
жиры=норма
то
коронарный_риск=средний.

правило26: если
холестерин=излишек
то
коронарный_риск=выше_среднего.

правило27: если
холестерин=норма
то
коронарный_риск=средний.

правило28: если
соль=излишек
то
кровяное_давление=выше_среднего.

правило29: если
соль=норма
то
кровяное_давление=среднее.

правило30: если
кальций=излишек
то
риск_остеопороза=ниже_среднего.

правило31: если
кальций=норма
то
риск_остеопороза=средний.

правило32: если
кальций=низкий
то
риск_остеопороза=выше_среднего.

правило33: если
относительный_вес=излишний и
коронарный_риск=выше_среднего и
кровяное_давление=выше_среднего и
курение=да
то
перспектива=унылая.

правило34: если
относительный_вес=излишний и
коронарный_риск=выше_среднего и
кровяное_давление=выше_среднего и
курение=нет
то
перспектива=плохая.

правило34: если
относительный_вес =излишний и
коронарный_риск =выше_среднего и
кровяное_давление =выше_среднего и
курение =нет
то
перспектива =плохая.

правило35: если
относительный_вес =излишний и
коронарный_риск =выше_среднего и
кровяное_давление =среднее и
курение =нет
то
перспектива =хорошая.

правило36: если
относительный_вес =излишний и
сердзаб =низкий и
кровяное_давление =среднее и
курение =нет
то
перспектива =хорошая.

правило37: если
зп =да и
сердзаб =низкий и
кровяное_давление =среднее и
курение =нет
то
перспектива =отличная.

правило38: если
зп =да и
коронарный_риск =выше_среднего и
кровяное_давление =среднее и
курение =да
то
перспектива =посредственная.

правило39: если
зп=да и
сердзаб=низкий и
кровяное_давление=среднее и
курение=да
то
перспектива=посредственная.

правило40: если
зп=да и
сердзаб=низкий и
кровяное_давление=среднее и
курение=нет
то
перспектива=хорошая.

правило41: если
старт=да
то
перспектива=неизвестна.

правило42: если
раса=негроидная и
происхождение=средиземноморское
то
риск=высокий.

правило43: если
характер=агрессивный
то
тип_личности=тип_а.

правило44: если
характер=мягкий
то
тип_личности=тип_b.

правило45: если
тип_личности=тип_а
то
риск=высокий.

```
правило46: если
        старт=да
то
        риск=неизвестен.

правило47: если
        потребление_алкоголя=умеренное
то
        дополн=посредственно.

правило48: если
        потребление_алкоголя=не_употребляет
то
        дополн=хорошо.

правило49: если
        потребление_алкоголя=чрезмерно
то
        дополн=плохо.

правило50: если
        перспектива=унылая и
        риск=высокий и
        дополн=плохо
то
        фактор=минус_12.

правило51: если
        перспектива=отличная и
        риск=неизвестен и
        дополн=посредственно
то
        фактор=плюс_12.

правило52: если
        перспектива=отличная и
        риск=высокий и
        дополн=посредственно
то
        фактор=ноль.
```

правило53: если
перспектива =хорошая и
риск =высокий и
дополн =посредственно
то
фактор=поль.

правило54: если
перспектива =хорошая и
риск =высокий
то
фактор=минус_12.

правило55: если
перспектива =посредственная и
риск =высокий
то
фактор=минус_12.

правило56: если
перспектива =отличная и
риск =неизвестен и
дополн =хорошо
то
фактор=плюс_12.

правило57: если
перспектива =отличная и
риск =неизвестен и
дополн =плохо
то
фактор=плюс_12.

правило58: если
перспектива =посредственная и
риск =неизвестен
то
фактор=плюс_12.

правило59: если
перспектива=хорошая и
риск=неизвестен
то
фактор=ноль.

правило60: если
перспектива=унылая и
риск=неизвестен
то
фактор=ноль.

правило61: если
старт=да
то
фактор=неизвестен.

правило62: если
старт=выход
то
фактор=выход.

правило63: если
основная_продолжительность=72 и
фактор=ноль
то
продолжительность=72_года.

правило64: если
основная_продолжительность=67 и
фактор=ноль
то
продолжительность=67_лет.

правило65: если
основная_продолжительность=64 и
фактор=ноль
то
продолжительность=64_года.

правило66: если
основная_продолжительность=62 и
фактор=пол
то
продолжительность=62_года.

правило67: если
основная_продолжительность=60 и
фактор=пол
то
продолжительность=60_лет.

правило68: если
основная_продолжительность=72 и
фактор=плюс_12
то
продолжительность=84_года.

правило69: если
основная_продолжительность=67 и
фактор=плюс_12
то
продолжительность=79_лет.

правило70: если
основная_продолжительность=64 и
фактор=плюс_12
то
продолжительность=76_лет.

правило71: если
основная_продолжительность=62 и
фактор=плюс_12
то
продолжительность=74_года.

правило72: если
основная_продолжительность=60 и
фактор=плюс_12
то
продолжительность=72_года.

правило73: если
основная_продолжительность=72 и
фактор=минус_12
то
продолжительность=60_лет.

правило74: если
основная_продолжительность=67 и
фактор=минус_12
то
продолжительность=55_лет.

правило75: если
основная_продолжительность=64 и
фактор=минус_12
то
продолжительность=52_года.

правило76: если
основная_продолжительность=62 и
фактор=минус_12
то
продолжительность=50_лет.

правило77: если
основная_продолжительность=60 и
фактор=минус_12
то
продолжительность=48_лет.

Пример работы с базой данных оценки состояния здоровья

Какова ваша цель?

(Введите(ОБЪЕКТ))

продолжительность

Вы готовы начать?
1. да
2. выход
1

Сколько испытуемому лет?
1. 25_или_меньше
2. 25-55
3. 55_или_больше
2

Какого пола испытуемый?
1. м
2. ж
2

Каков вес испытуемого?
1. 55_или_меньше
2. 55-85
3. 85_или_больше
1

Телосложение:
1. мелкое
2. крупное
1

Потребление ненасыщенных жиров:
1. норма
2. излишек
1

Потребление соли:
1. норма
2. излишек
1

Испытуемый курит?
1. да
2. нет
2

Какой расы испытуемый?
1. негроидная
2. европеоидная
3. монголоидная
3

Характер испытуемого:
1. агрессивный
2. мягкий
1

Потребление алкоголя:
1. не_употребляет
2. умеренно
3. чрезмерно
2

РЕЗУЛЬТАТ КОНСУЛЬТАЦИИ
продолжительность=67_лет

КОНЕЦ КОНСУЛЬТАЦИИ

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ

Научное изданне

Бриан Сойер
Деннис **Л. Фостер**

ПРОГРАММИРОВАНИЕ ЭКСПЕРТНЫХ СИСТЕМ НА ПАСКАЛЕ

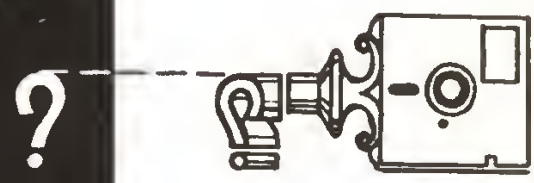
Книга одобрена на заседании секции редсовета
по электрониой обработке данных в экономике
15.12.87

Зав. редакцией *К. В. Коробов*
Редактор *О. А. Ермилина*
Худож. редактор *Ю. И. Артюхов*
Корректор *Т. М. Васильева*
Обложка художнпка *О. В. Кривцова*
ИБ № 2433

Подпнсано в печать 24.08.89.
Формат 60 × 88$^{1}/_{16}$. Бум. офсетная. Гарнитура «Таймс».
Печать офсетная. Усл. п. л. 11,76. Усл. кр.-отт. 12,25. Уч.-изд. л. 7,6.
Тираж 40 000 экз. Заказ 852. Цена 55 коп.

Издательство «Финансы и статнстнка»,
101000, Москва, ул. Чернышевского, 7.
Тнпография нм. Котлякова издательства «Финансы н статнстика»
Государственного комнтета СССР по печатн,
195273, Леннград, ул. Руставелн, 13.

Искусственный интеллект стал достоянием пользователей персональных ЭВМ. Экспертные системы перестали быть областью чисто академических исследований и превратились в один из самых популярных продуктов на рынке программного обеспечения. "Программирование экспертных систем на Паскале" — одно из первых практических руководств по планированию, разработке и программированию экспертных систем для ПЭВМ, основанных на знаниях. Книга предлагает вашему вниманию написанную на Паскале экспертную систему, разработанную Брианом Сойером и Дэнисом Фостером.



55 коп.